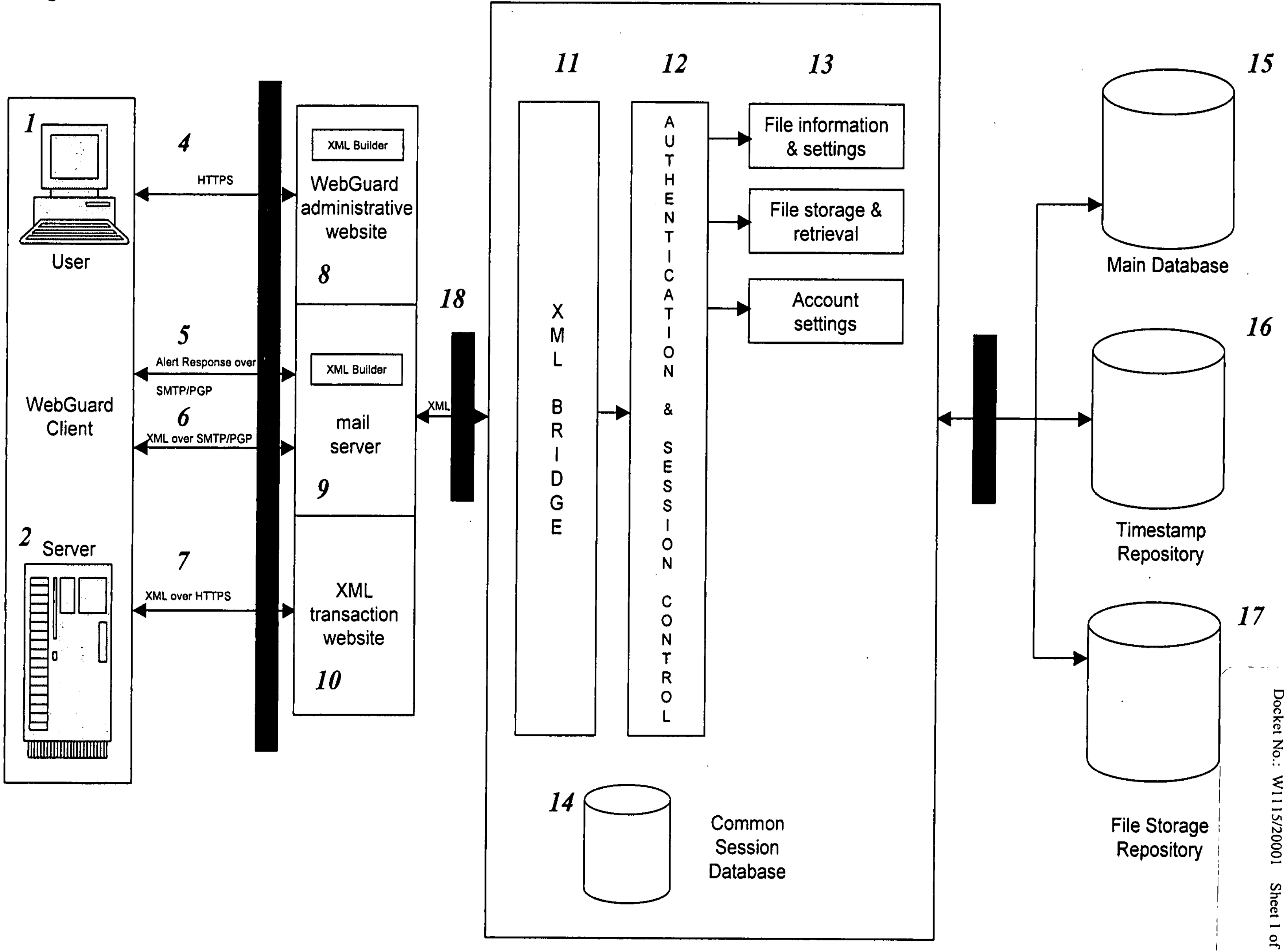
Applicants: Matthew J. Dye, et al.
Filed: November 20, 2003 Customer No. 03000
For: REMOTE WEB SITE SECURITY SYSTEM
Docket No.: W1115/20001 Sheet 1 of 30

Figure 1: WebGuard Remote Administrative Architecture



Applicants: Matthew J. Dye, et al.
Filed: November 20, 2003 Customer No. 03000
For: REMOTE WEB SITE SECURITY SYSTEM
Docket No.: W1115/20001 Sheet 2 of 30

Figure 2: Simple Remote Process Overview

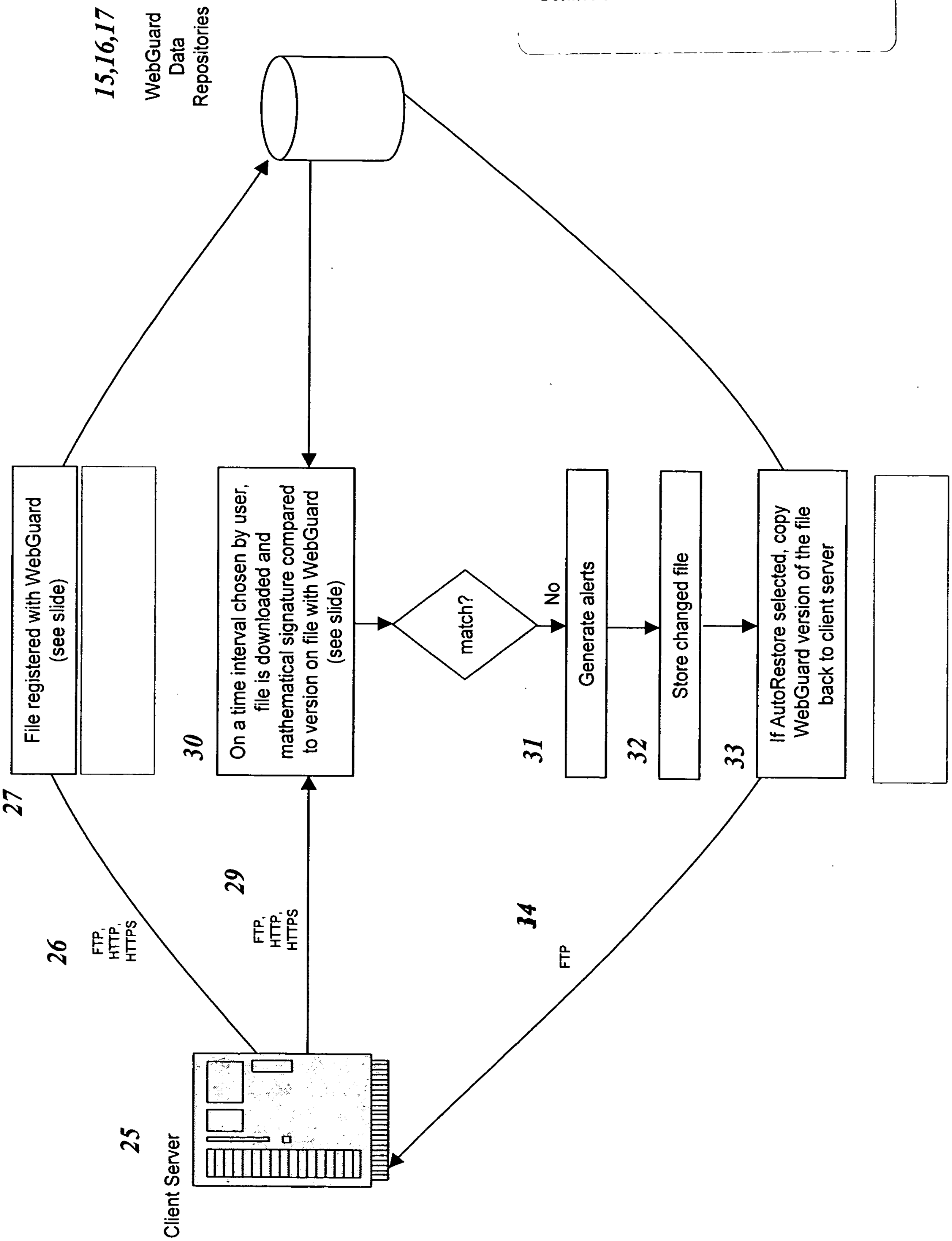

15,16,17
WebGuard Data Repositories
25
Client Server
26
FTP, HTTP, HTTPS
27
File registered with WebGuard (see slide)
29
FTP, HTTP, HTTPS
30
On a time interval chosen by user, file is downloaded and mathematical signature compared to version on file with WebGuard (see slide)
match?
No
31
Generate alerts
32
Store changed file
33
If AutoRestore selected, copy WebGuard version of the file back to client server
34
FTP

Figure 3: File Registration

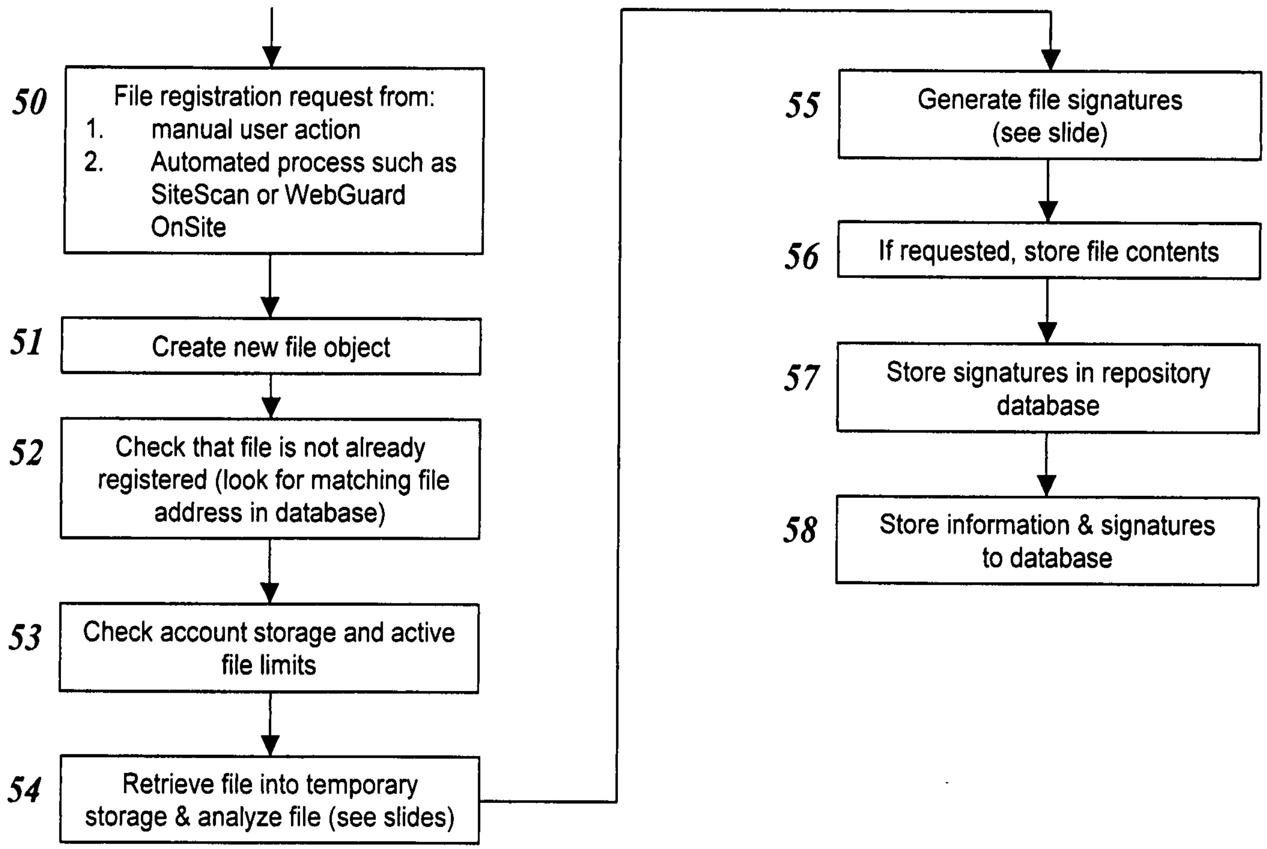

50
File registration request from:
1. manual user action
2. Automated process such as SiteScan or WebGuard OnSite
51
Create new file object
52
Check that file is not already registered (look for matching file address in database)
53
Check account storage and active file limits
54
Retrieve file into temporary storage & analyze file (see slides)
55
Generate file signatures (see slide)
56
If requested, store file contents
57
Store signatures in repository database
58
Store information & signatures to database

Figure 4: HTTP(S) File Retrieval

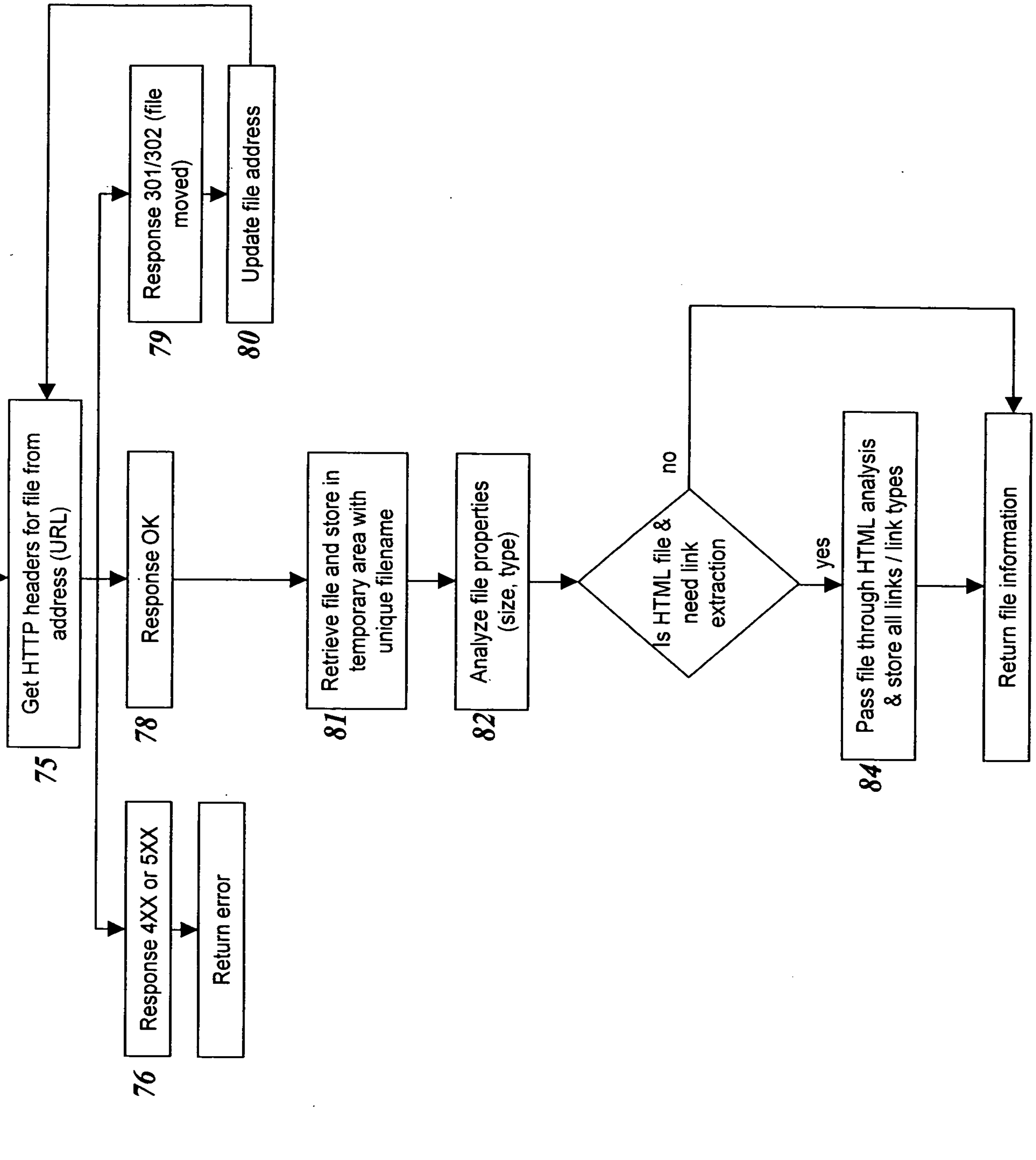

Get HTTP headers for file from address (URL)
75
Response 301/302 (file moved)
79
Update file address
80
Response OK
78
Response 4XX or 5XX
76
Return error
Retrieve file and store in temporary area with unique filename
81
Analyze file properties (size, type)
82
Is HTML file & need link extraction
no
yes
Pass file through HTML analysis & store all links / link types
84
Return file information

Figure 5: FTP File Retrieval

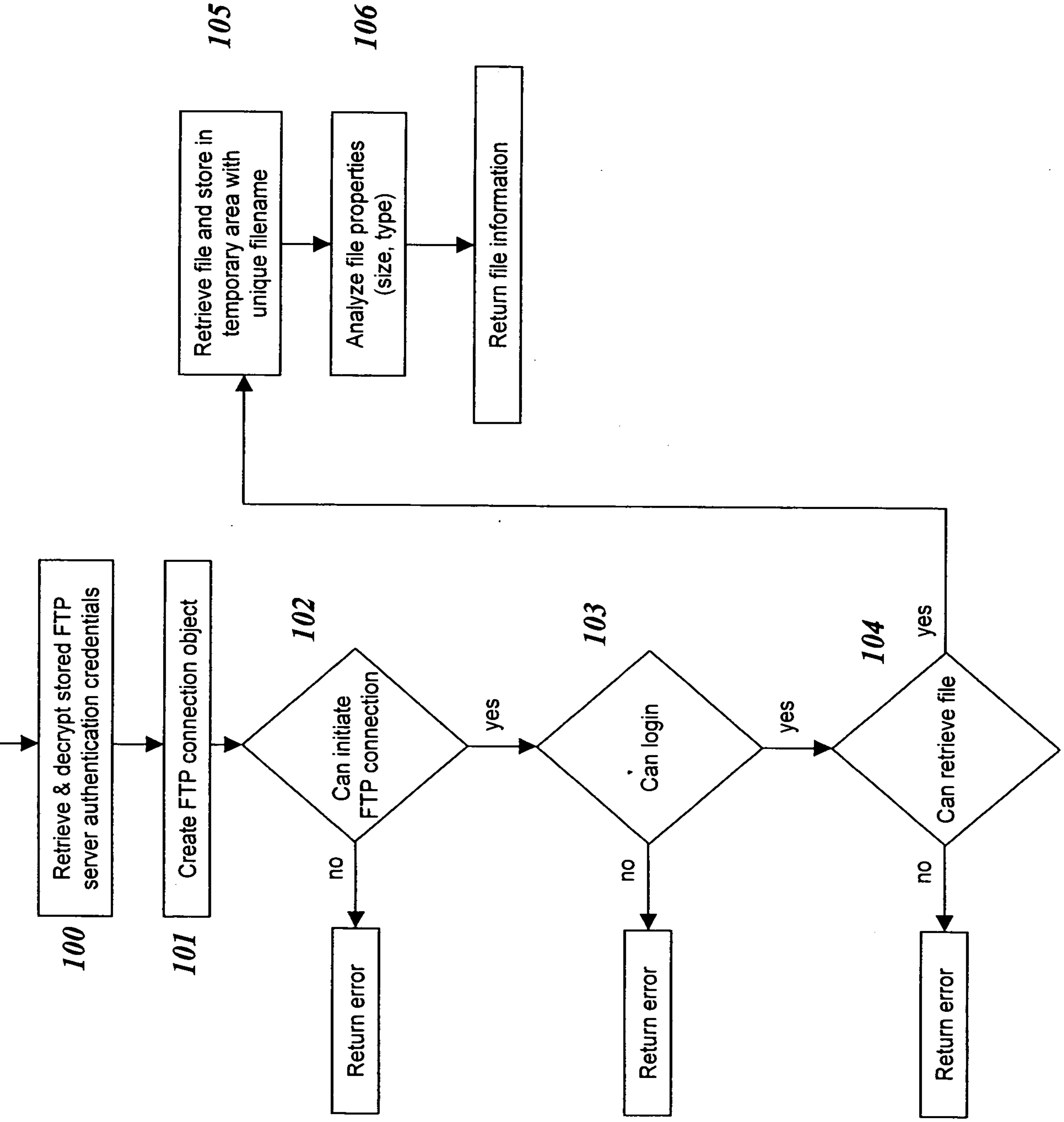

100
Retrieve & decrypt stored FTP server authentication credentials
101
Create FTP connection object
102
Can initiate FTP connection
no
Return error
yes
103
Can login
no
Return error
yes
104
Can retrieve file
no
Return error
yes
105
Retrieve file and store in temporary area with unique filename
106
Analyze file properties (size, type)
Return file information

Figure 6: File Signature Process

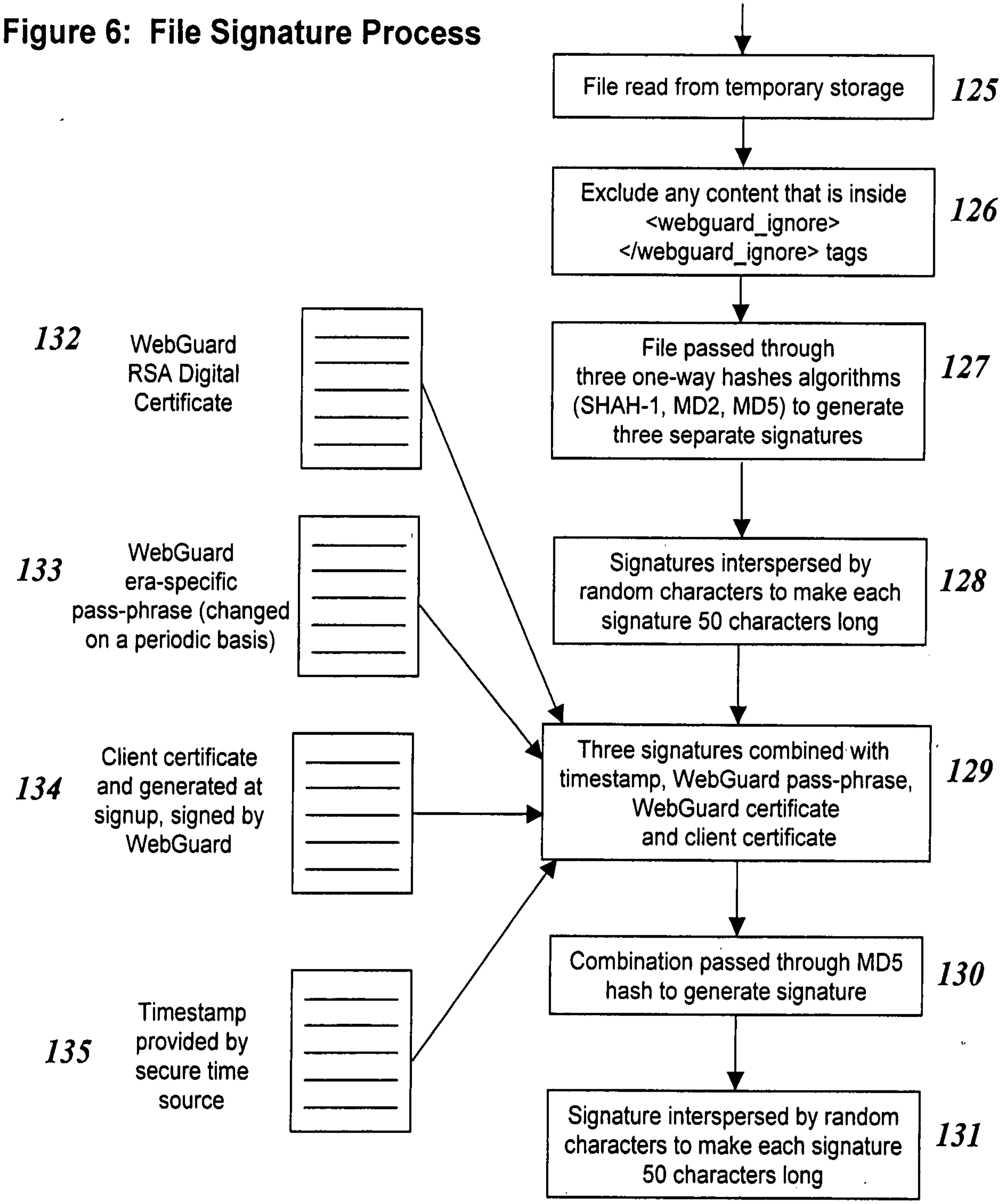

File read from temporary storage
125
Exclude any content that is inside <webguard_ignore> </webguard_ignore> tags
126
File passed through three one-way hashes algorithms (SHAH-1, MD2, MD5) to generate three separate signatures
127
Signatures interspersed by random characters to make each signature 50 characters long
128
Three signatures combined with timestamp, WebGuard pass-phrase, WebGuard certificate and client certificate
129
Combination passed through MD5 hash to generate signature
130
Signature interspersed by random characters to make each signature 50 characters long
131
132
WebGuard RSA Digital Certificate
133
WebGuard era-specific pass-phrase (changed on a periodic basis)
134
Client certificate and generated at signup, signed by WebGuard
135
Timestamp provided by secure time source

Figure 7: File Storage

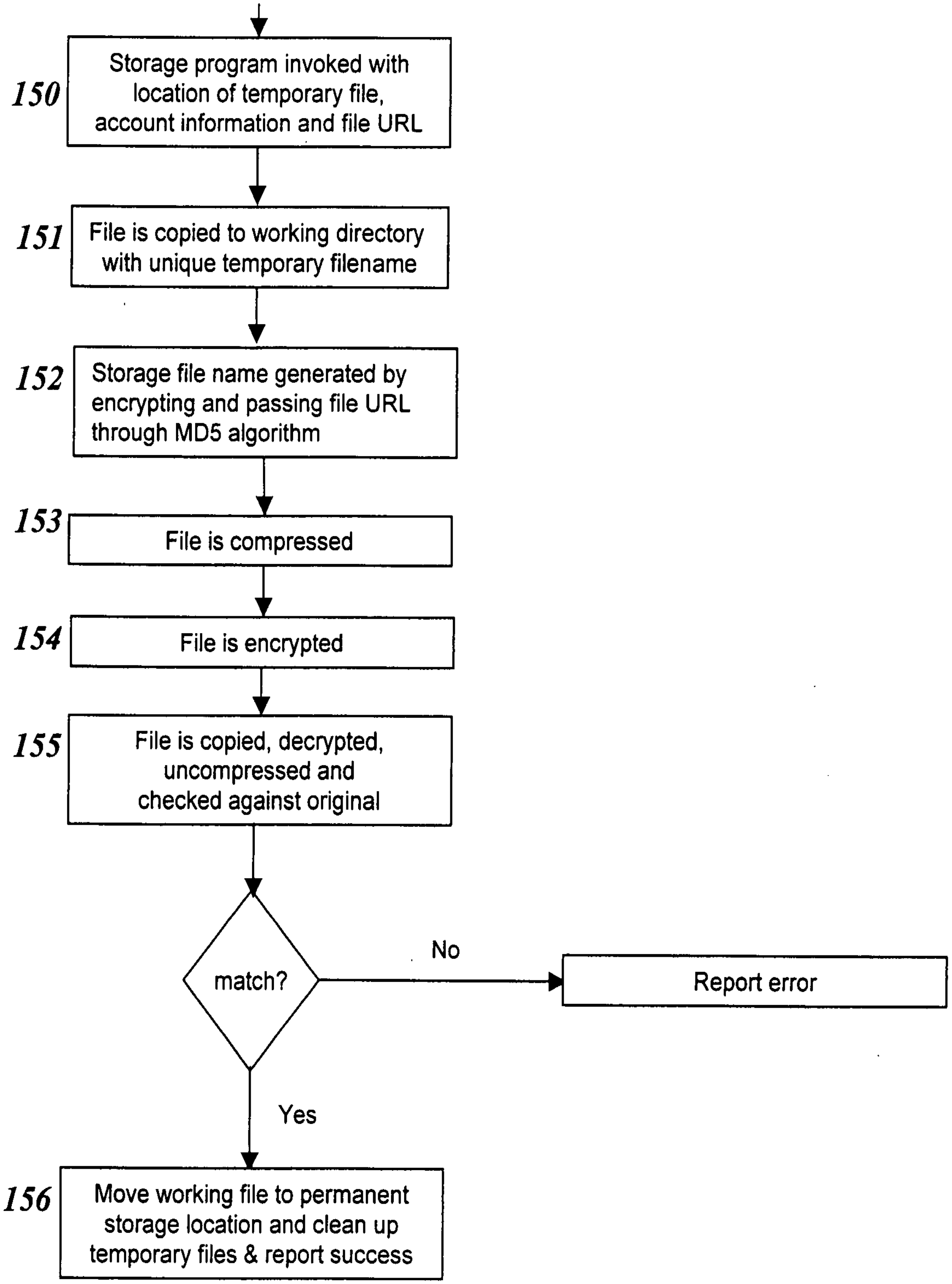

150
Storage program invoked with location of temporary file, account information and file URL
151
File is copied to working directory with unique temporary filename
152
Storage file name generated by encrypting and passing file URL through MD5 algorithm
153
File is compressed
154
File is encrypted
155
File is copied, decrypted, uncompressed and checked against original
match?
No
Report error
Yes
156
Move working file to permanent storage location and clean up temporary files & report success

Figure 8: HTTP/HTTPS File Scan Launcher

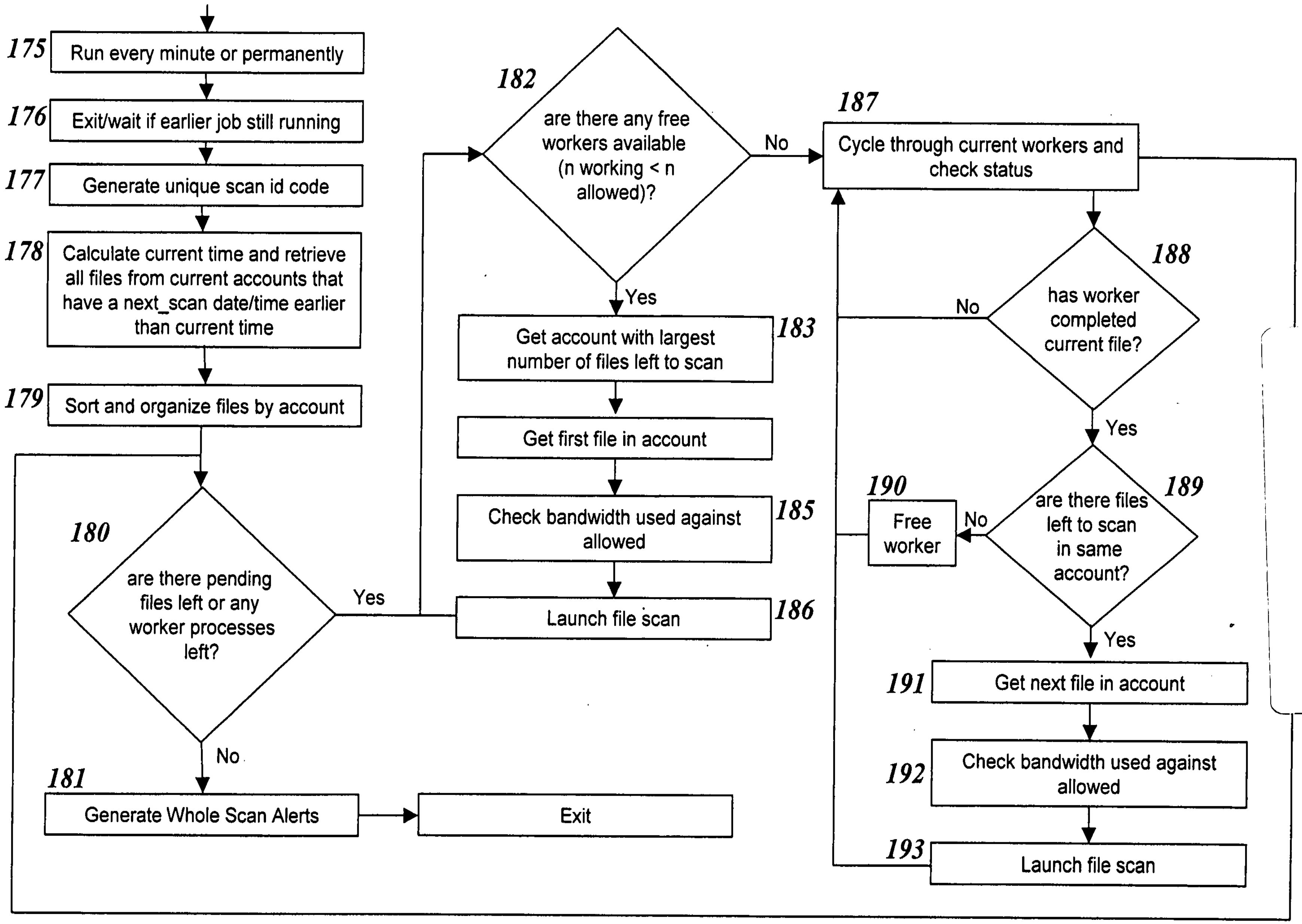

175 Run every minute or permanently
176 Exit/wait if earlier job still running
177 Generate unique scan id code
178 Calculate current time and retrieve all files from current accounts that have a next_scan date/time earlier than current time
179 Sort and organize files by account
180 are there pending files left or any worker processes left?
Yes
No
181 Generate Whole Scan Alerts
Exit
182 are there any free workers available (n working < n allowed)?
Yes
No
183 Get account with largest number of files left to scan
Get first file in account
185 Check bandwidth used against allowed
186 Launch file scan
187 Cycle through current workers and check status
188 has worker completed current file?
No
Yes
189 are there files left to scan in same account?
No
Yes
190 Free worker
191 Get next file in account
192 Check bandwidth used against allowed
193 Launch file scan

Figure 9: FTP File Scan Launcher

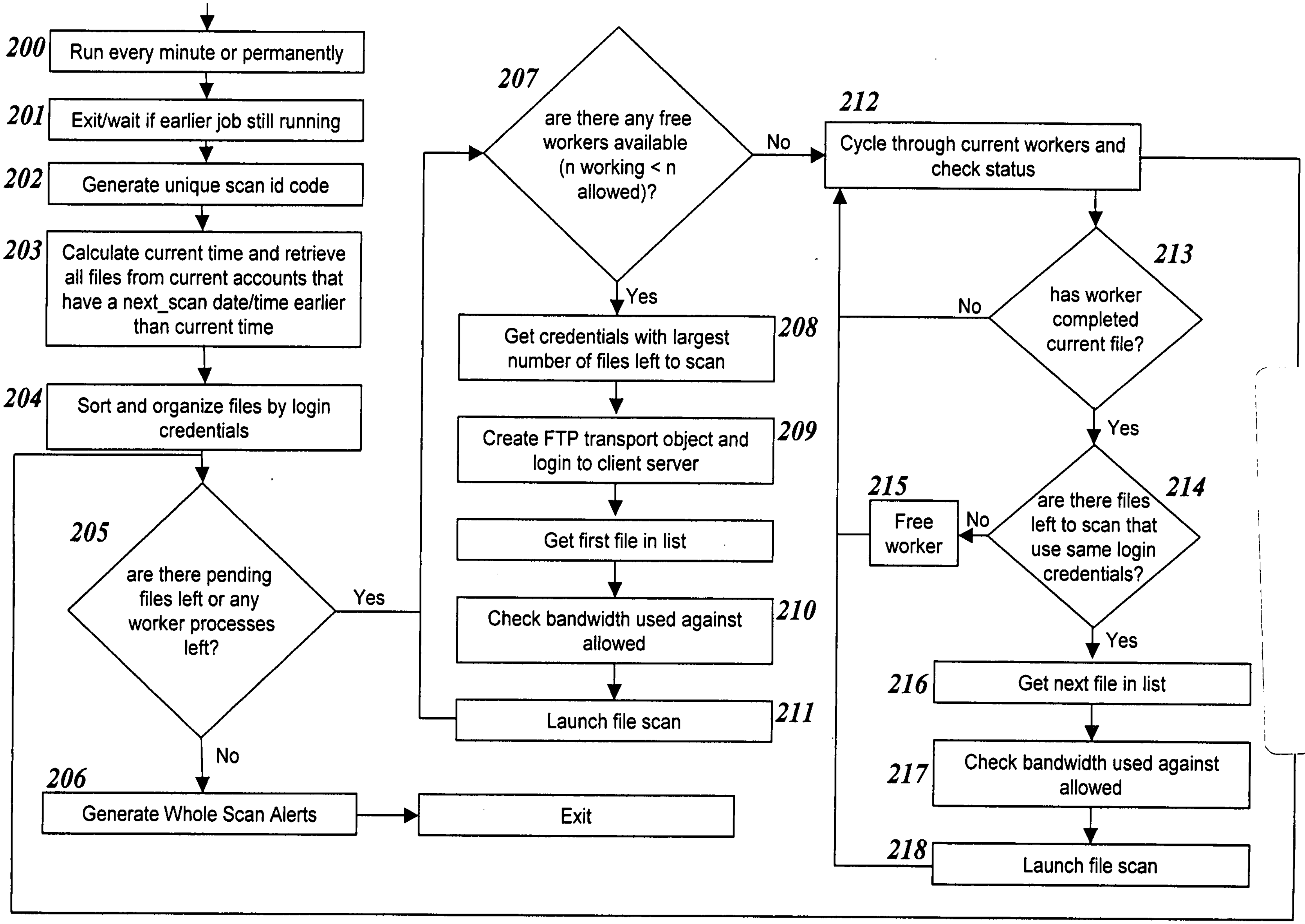

200 Run every minute or permanently
201 Exit/wait if earlier job still running
202 Generate unique scan id code
203 Calculate current time and retrieve all files from current accounts that have a next_scan date/time earlier than current time
204 Sort and organize files by login credentials
205 are there pending files left or any worker processes left?
Yes
No
206 Generate Whole Scan Alerts
Exit
207 are there any free workers available (n working < n allowed)?
Yes
No
208 Get credentials with largest number of files left to scan
209 Create FTP transport object and login to client server
Get first file in list
210 Check bandwidth used against allowed
211 Launch file scan
212 Cycle through current workers and check status
213 has worker completed current file?
No
Yes
214 are there files left to scan that use same login credentials?
No
Yes
215 Free worker
216 Get next file in list
217 Check bandwidth used against allowed
218 Launch file scan

Figure 10: File Scan

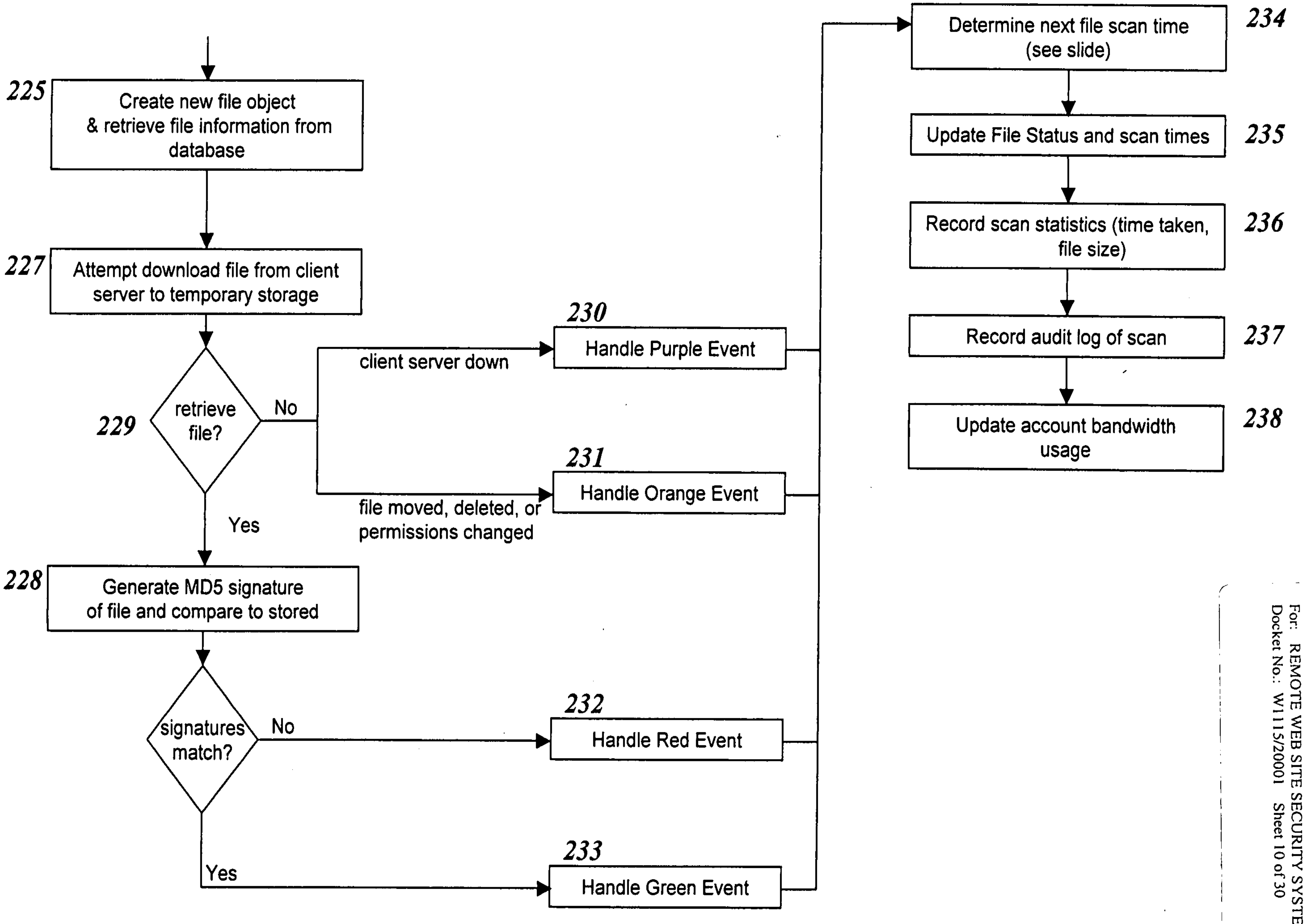

225
Create new file object & retrieve file information from database
227
Attempt download file from client server to temporary storage
229
retrieve file?
No
client server down
file moved, deleted, or permissions changed
Yes
228
Generate MD5 signature of file and compare to stored
signatures match?
No
Yes
230
Handle Purple Event
231
Handle Orange Event
232
Handle Red Event
233
Handle Green Event
234
Determine next file scan time (see slide)
235
Update File Status and scan times
236
Record scan statistics (time taken, file size)
237
Record audit log of scan
238
Update account bandwidth usage

Figure 11: Determining Next File Scan Time

- WebGuard allows users to choose the scan interval (e.g. every 30 minutes, 1 day, 12 hours) for each file but does not allow users to choose the time of day that each file is scanned. WebGuard reserves the right to assign those times in order to ensure efficient bandwidth usage.

- Each file is assigned a 'base time', which is a number of minutes past midnight. The next scan time is then calculated from the base time and the scan interval. i.e. if the base time is 30 minutes and the scan interval is 60 minutes, scans will happen at 12:30, 1:30, 2:30, 3:30 etc. This allows WebGuard to spread bandwidth usage and prevents creep in scan times if the file takes longer than anticipated to scan. i.e. if we simply used the scan interval without the base time, we could be running scans at 12:30, 1:31, 2:32, 3:33 etc.

- Times are calculated in Unix time (seconds since January 1st 1970 00:00:00)

Step 1 – determine base_time_today for this file's base time

base_time_today = time_at_last_midnight + base_time_in_seconds

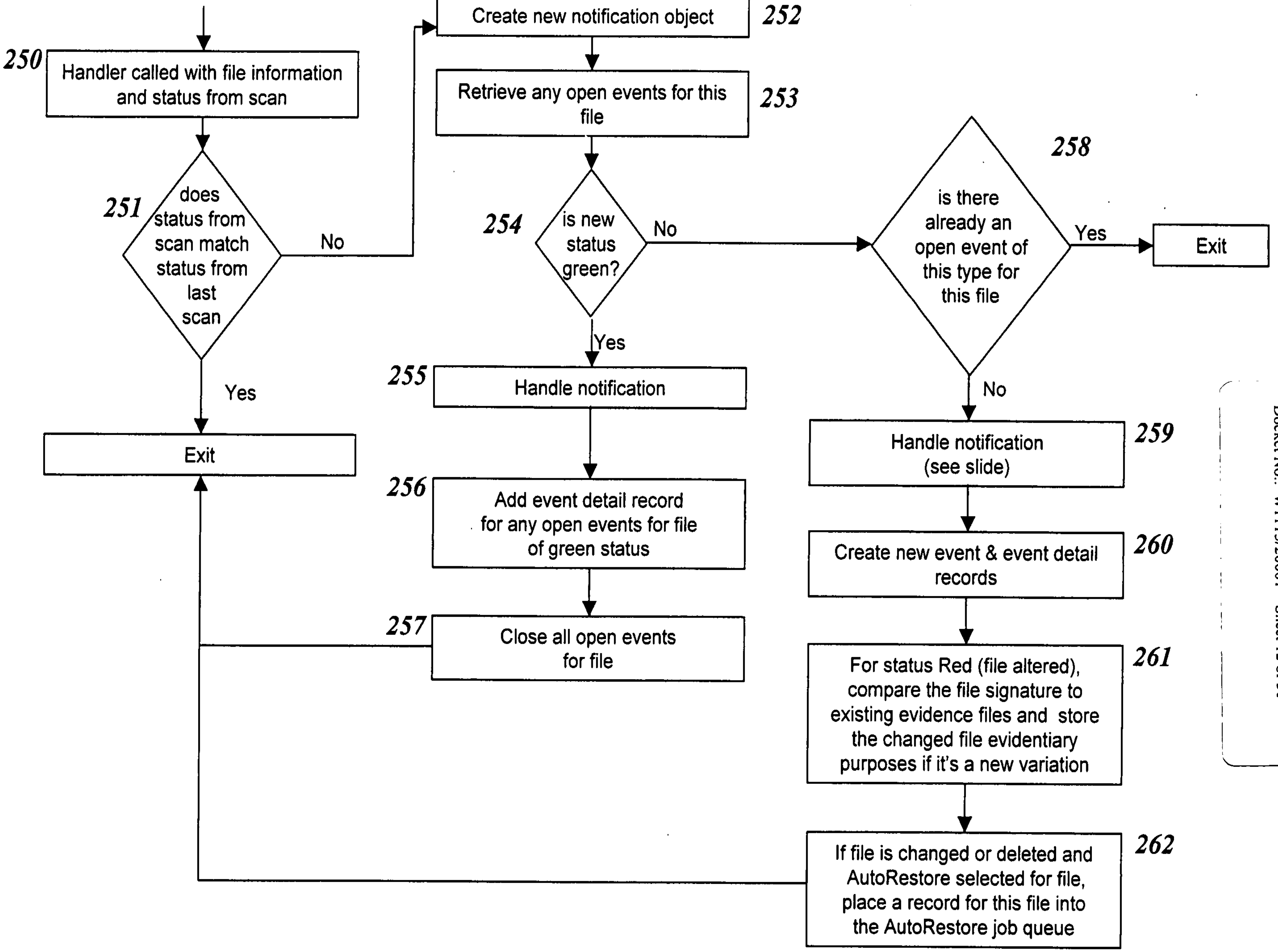
time_at_last_midnight is the time in seconds at midnight preceding the start of the scan
base_time_in_seconds is the base time for the file in seconds after midnight

Step 2 – determine the next scan time in Unix time

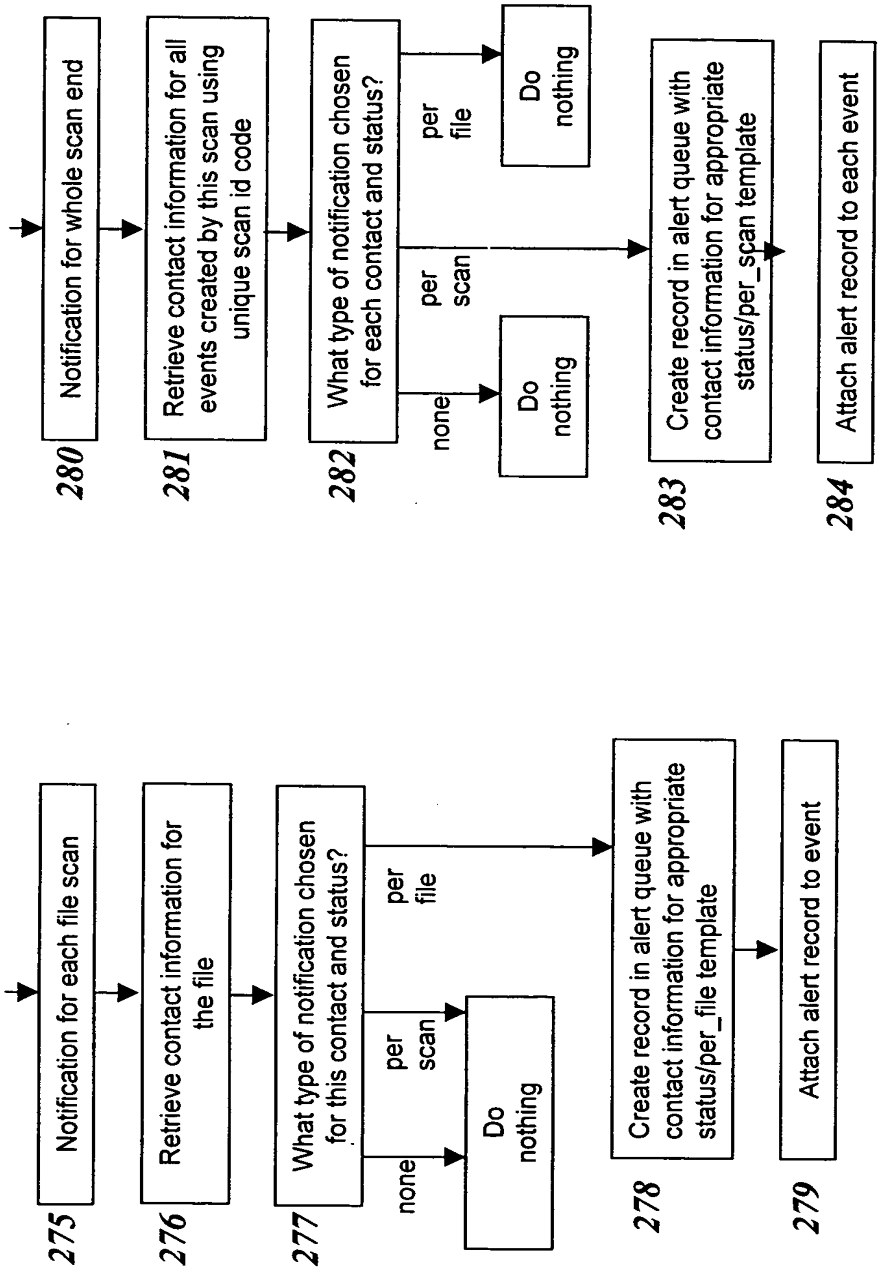
next scan time = base_time_today + ((integer (scan_start_time - base_time_today / scan_interval_in_seconds) + 1) * scan_interval_in_seconds

scan_start_time is the start of the scan in Unix time
scan_interval_in_seconds is the scan interval chosen by the user in seconds

Figure 12: Event Handler


250
Handler called with file information and status from scan
251
does status from scan match status from last scan
No
Yes
Exit
252
Create new notification object
253
Retrieve any open events for this file
254
is new status green?
No
Yes
255
Handle notification
256
Add event detail record for any open events for file of green status
257
Close all open events for file
258
is there already an open event of this type for this file
Yes
Exit
No
259
Handle notification (see slide)
260
Create new event & event detail records
261
For status Red (file altered), compare the file signature to existing evidence files and store the changed file evidentiary purposes if it's a new variation
262
If file is changed or deleted and AutoRestore selected for file, place a record for this file into the AutoRestore job queue

Applicants: Matthew J. Dye, et al.
Filed: November 20, 2003 Customer No. 03000
For: REMOTE WEB SITE SECURITY SYSTEM
Docket No.: W1115/20001 Sheet 13 of 30



280
Notification for whole scan end
281
Retrieve contact information for all events created by this scan using unique scan id code
282
What type of notification chosen for each contact and status?
none
Do nothing
per scan
per file
Do nothing
283
Create record in alert queue with contact information for appropriate status/per_scan template
284
Attach alert record to each event
275
Notification for each file scan
276
Retrieve contact information for the file
277
What type of notification chosen for this contact and status?
none
per scan
Do nothing
per file
278
Create record in alert queue with contact information for appropriate status/per_file template
279
Attach alert record to event

Figure 13: Notification Handler

Figure 14: AutoRestore

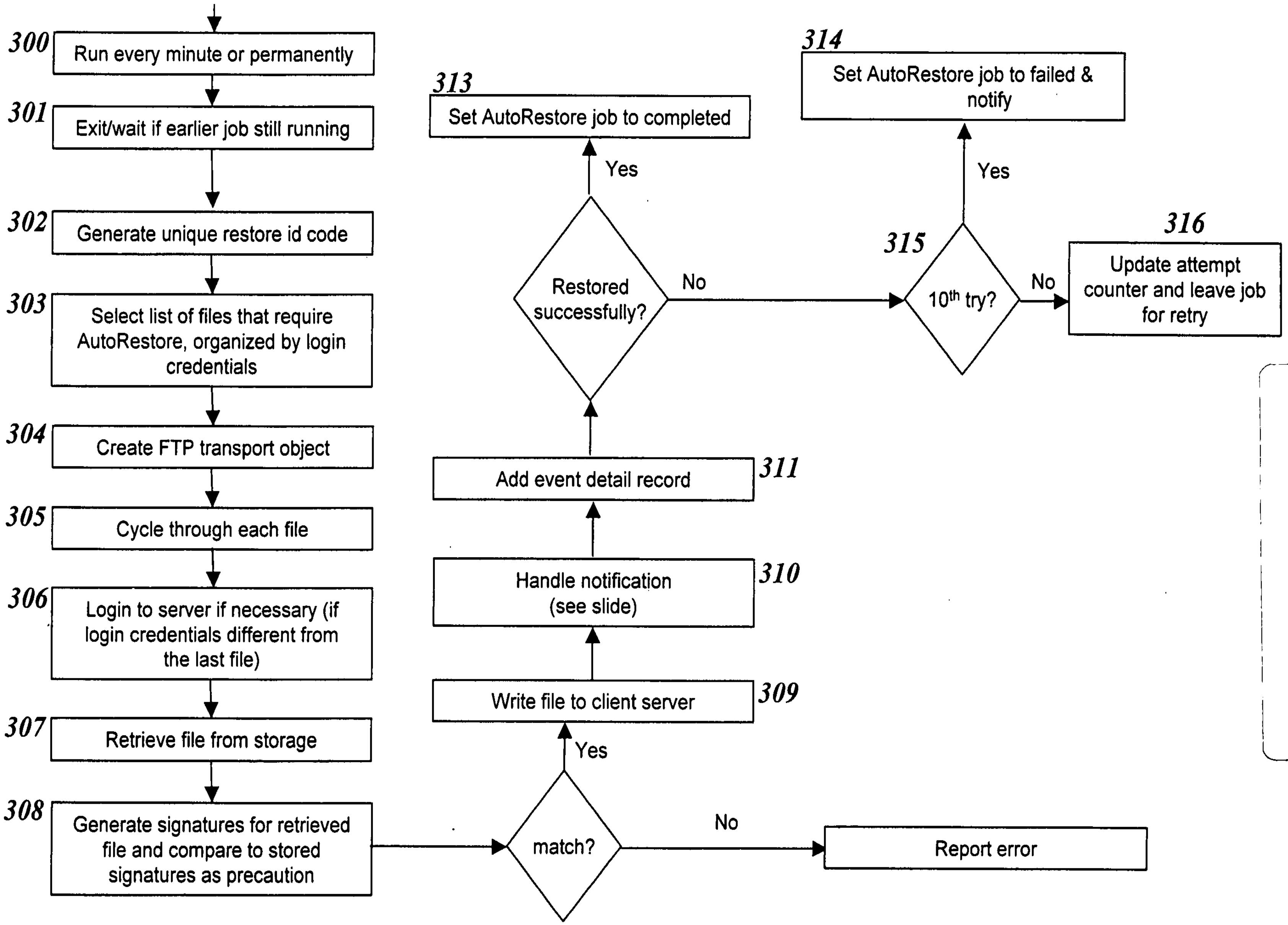

300 Run every minute or permanently
301 Exit/wait if earlier job still running
302 Generate unique restore id code
303 Select list of files that require AutoRestore, organized by login credentials
304 Create FTP transport object
305 Cycle through each file
306 Login to server if necessary (if login credentials different from the last file)
307 Retrieve file from storage
308 Generate signatures for retrieved file and compare to stored signatures as precaution
match?
Yes
No
Report error
309 Write file to client server
310 Handle notification (see slide)
311 Add event detail record
Restored successfully?
Yes
313 Set AutoRestore job to completed
No
315 10th try?
Yes
314 Set AutoRestore job to failed & notify
No
316 Update attempt counter and leave job for retry

Figure 15: SiteScan Launcher

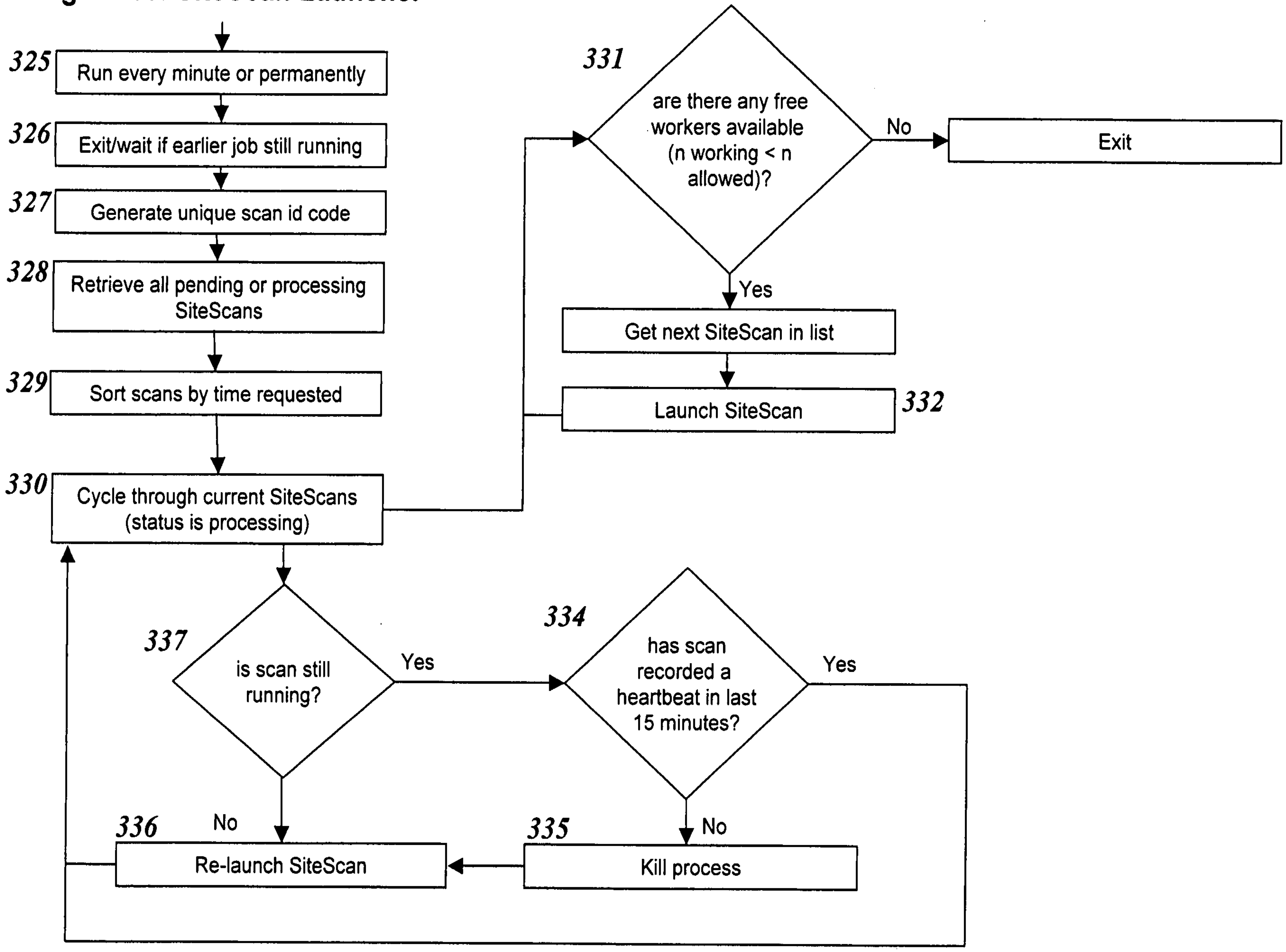

325
Run every minute or permanently
326
Exit/wait if earlier job still running
327
Generate unique scan id code
328
Retrieve all pending or processing SiteScans
329
Sort scans by time requested
330
Cycle through current SiteScans (status is processing)
331
are there any free workers available (n working < n allowed)?
No
Exit
Yes
Get next SiteScan in list
Launch SiteScan
332
337
is scan still running?
Yes
334
has scan recorded a heartbeat in last 15 minutes?
Yes
No
335
Kill process
No
336
Re-launch SiteScan

Figure 16: HTTP/HTTPS SiteScan

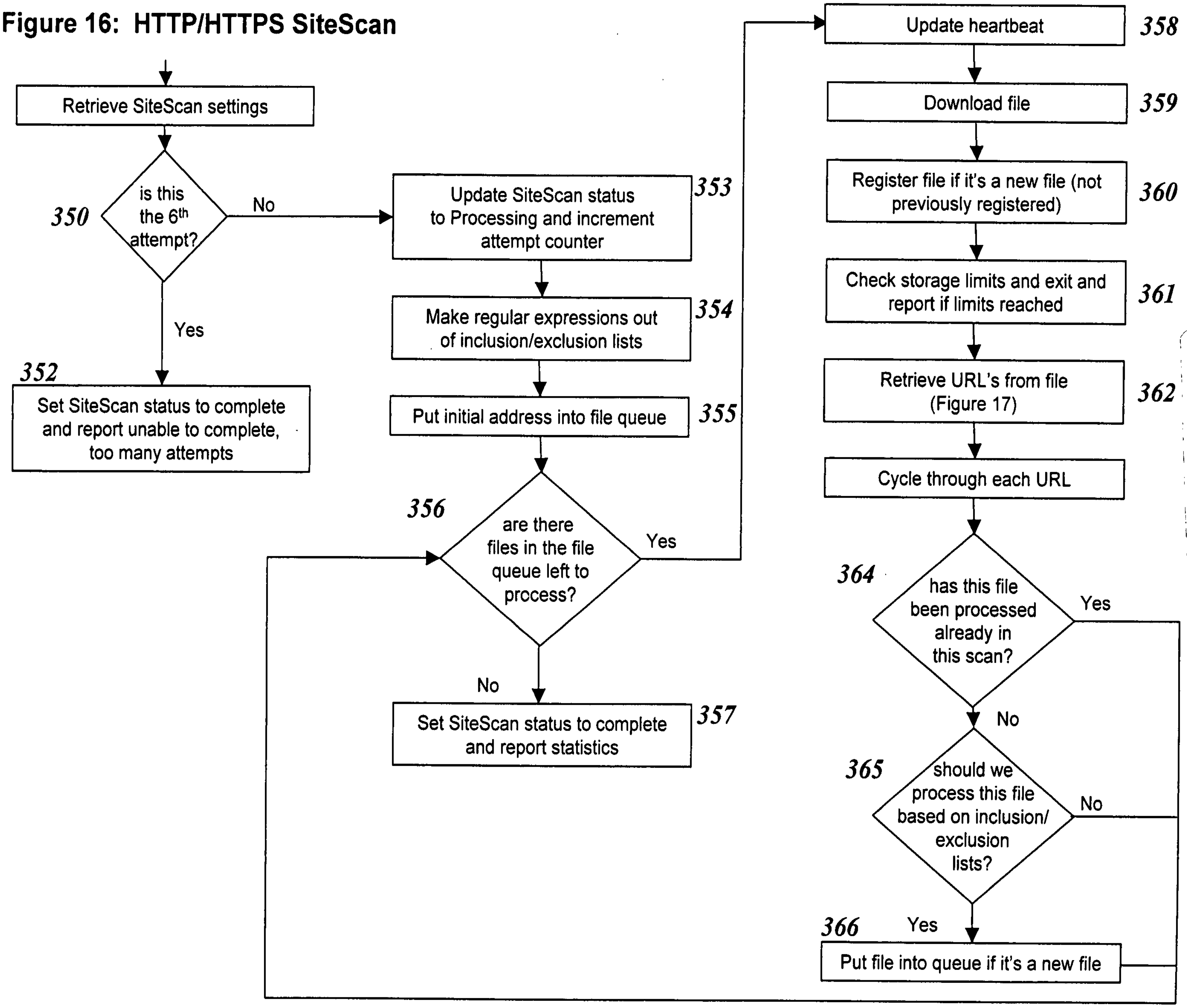

Retrieve SiteScan settings
350
is this the 6th attempt?
No
Yes
352
Set SiteScan status to complete and report unable to complete, too many attempts
Update SiteScan status to Processing and increment attempt counter
353
Make regular expressions out of inclusion/exclusion lists
354
Put initial address into file queue
355
356
are there files in the file queue left to process?
Yes
No
Set SiteScan status to complete and report statistics
357
Update heartbeat
358
Download file
359
Register file if it's a new file (not previously registered)
360
Check storage limits and exit and report if limits reached
361
Retrieve URL's from file (Figure 17)
362
Cycle through each URL
364
has this file been processed already in this scan?
Yes
No
365
should we process this file based on inclusion/ exclusion lists?
No
366
Yes
Put file into queue if it's a new file

Figure 17: Process a File to Extract URL's

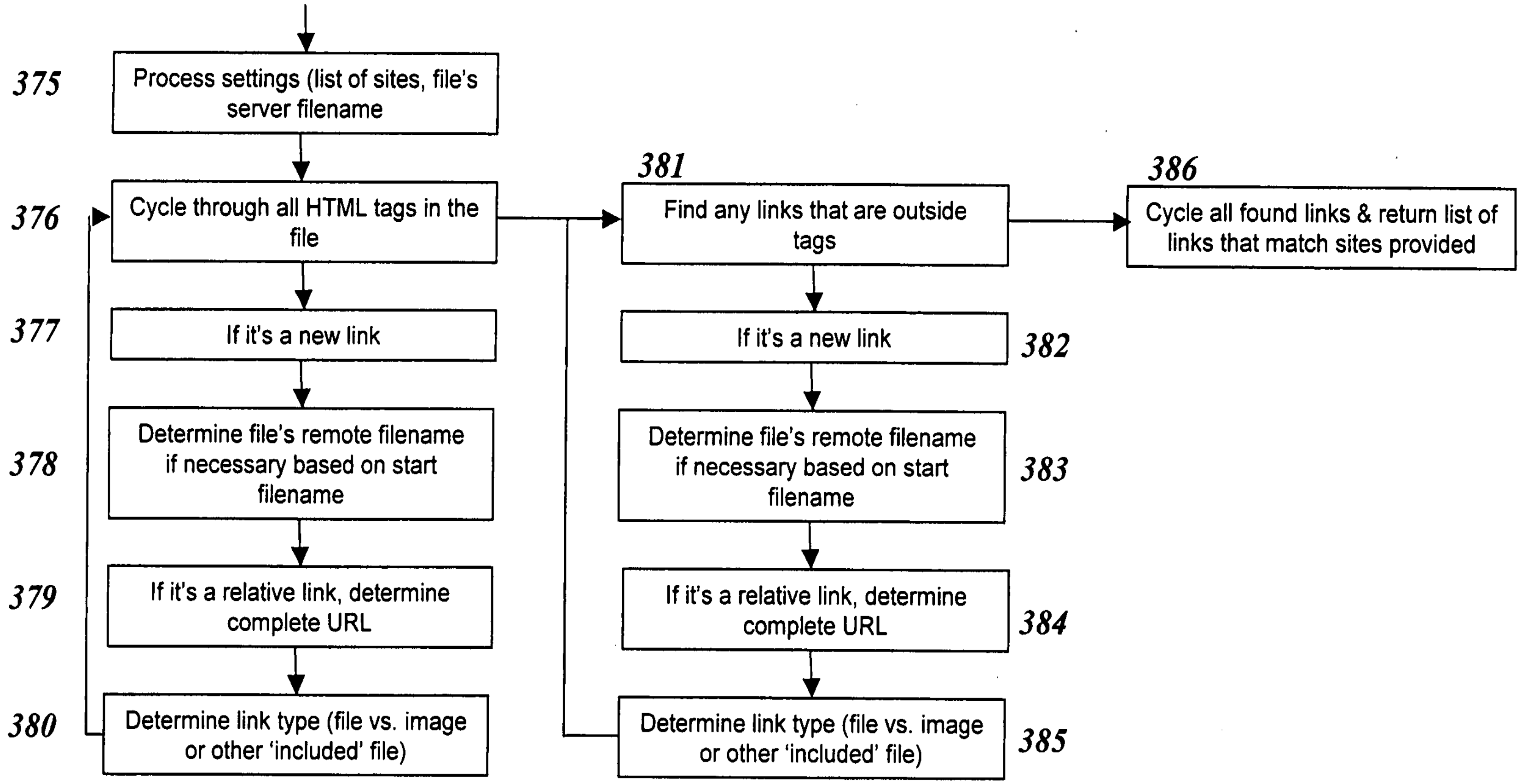

375
Process settings (list of sites, file's server filename
376
Cycle through all HTML tags in the file
377
If it's a new link
378
Determine file's remote filename if necessary based on start filename
379
If it's a relative link, determine complete URL
380
Determine link type (file vs. image or other 'included' file)
381
Find any links that are outside tags
382
If it's a new link
383
Determine file's remote filename if necessary based on start filename
384
If it's a relative link, determine complete URL
385
Determine link type (file vs. image or other 'included' file)
386
Cycle all found links & return list of links that match sites provided

Figure 18: FTP SiteScan

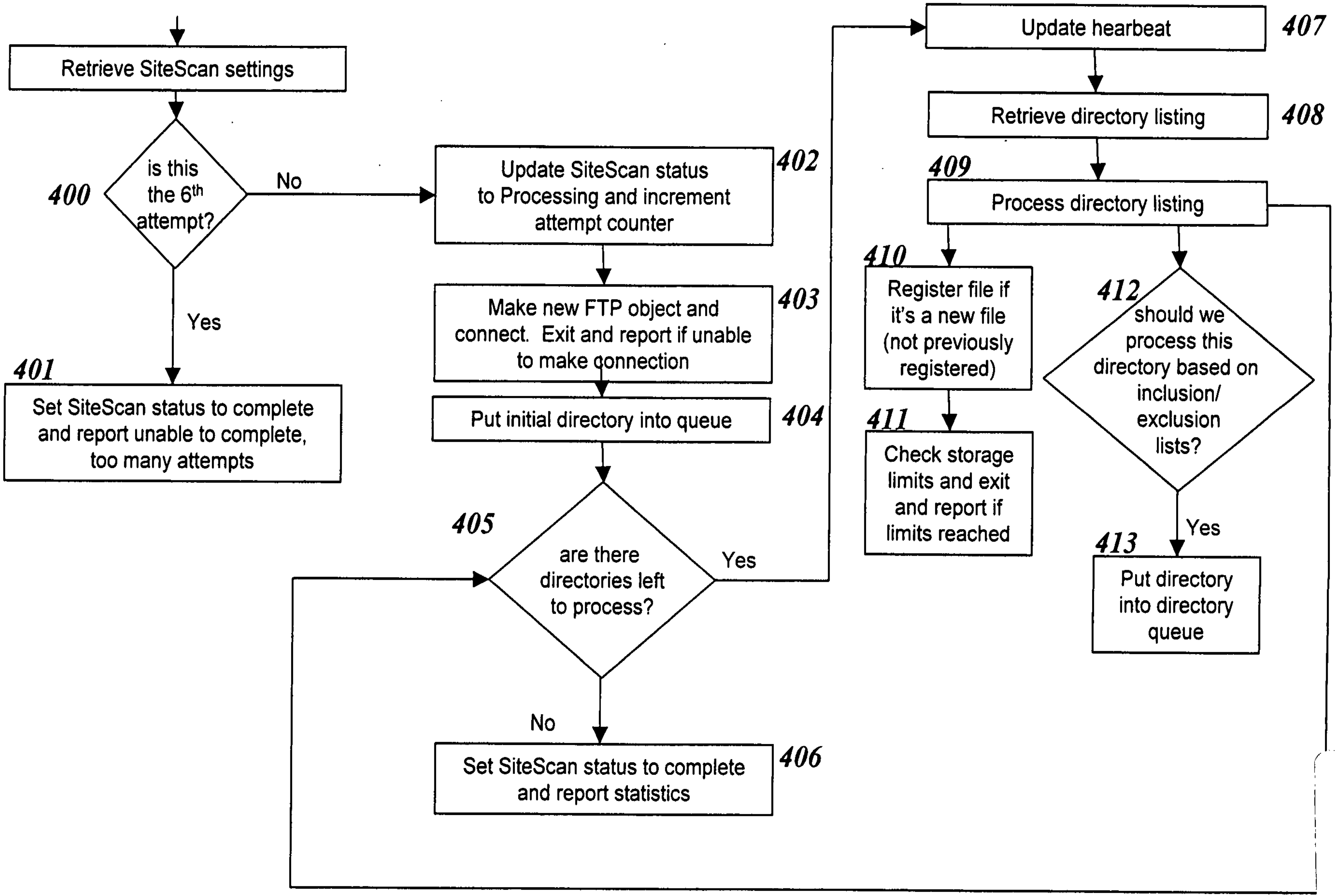

Retrieve SiteScan settings
400
is this the 6th attempt?
No
Yes
401
Set SiteScan status to complete and report unable to complete, too many attempts
402
Update SiteScan status to Processing and increment attempt counter
403
Make new FTP object and connect. Exit and report if unable to make connection
404
Put initial directory into queue
405
are there directories left to process?
Yes
No
406
Set SiteScan status to complete and report statistics
407
Update hearbeat
408
Retrieve directory listing
409
Process directory listing
410
Register file if it's a new file (not previously registered)
411
Check storage limits and exit and report if limits reached
412
should we process this directory based on inclusion/ exclusion lists?
Yes
413
Put directory into directory queue

Figure 19: Scan Log Compression

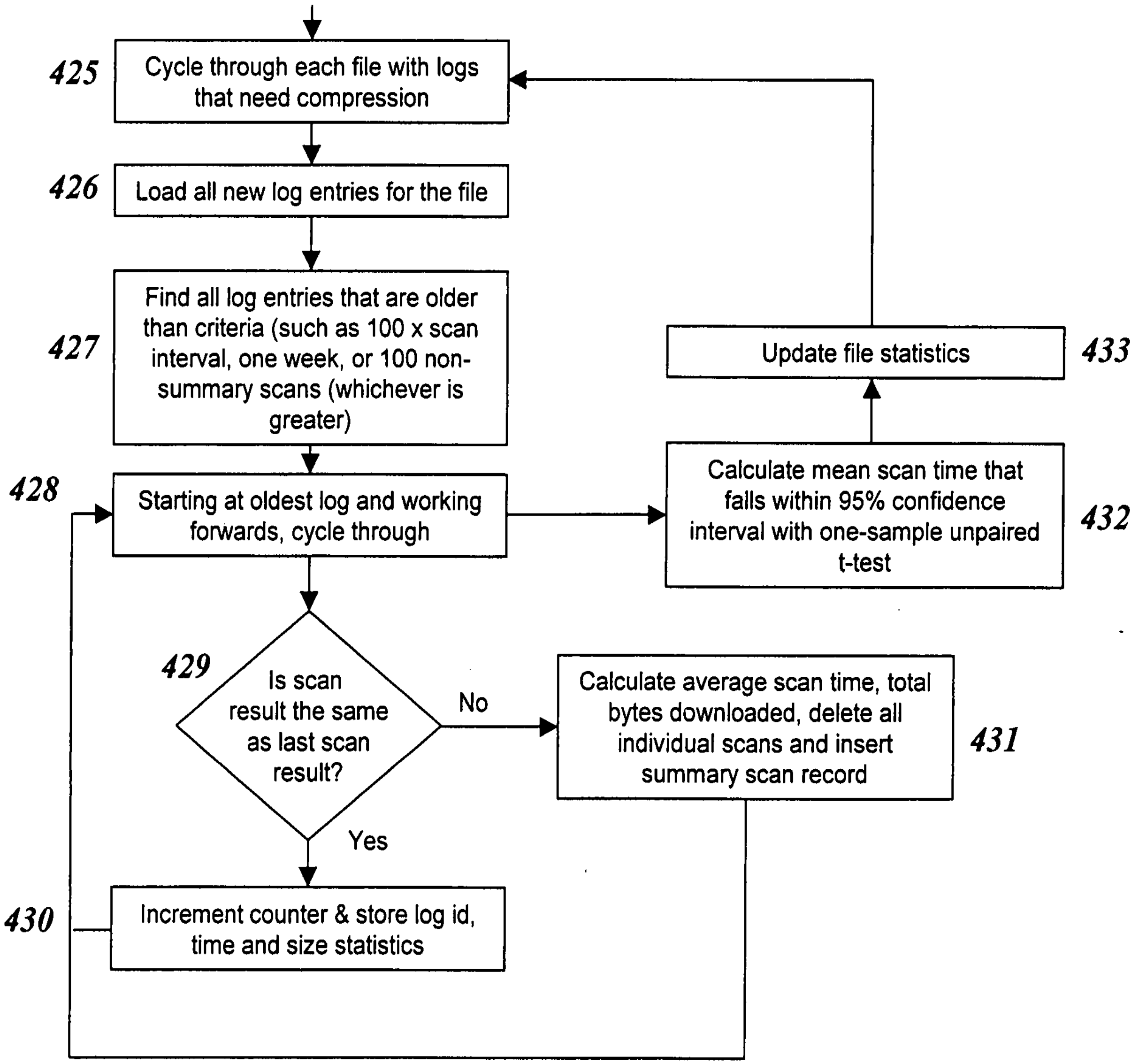

425
Cycle through each file with logs that need compression
426
Load all new log entries for the file
427
Find all log entries that are older than criteria (such as 100 x scan interval, one week, or 100 non-summary scans (whichever is greater)
428
Starting at oldest log and working forwards, cycle through
429
Is scan result the same as last scan result?
No
Yes
430
Increment counter & store log id, time and size statistics
431
Calculate average scan time, total bytes downloaded, delete all individual scans and insert summary scan record
432
Calculate mean scan time that falls within 95% confidence interval with one-sample unpaired t-test
433
Update file statistics

Figure 20: File Base-Time Assignment

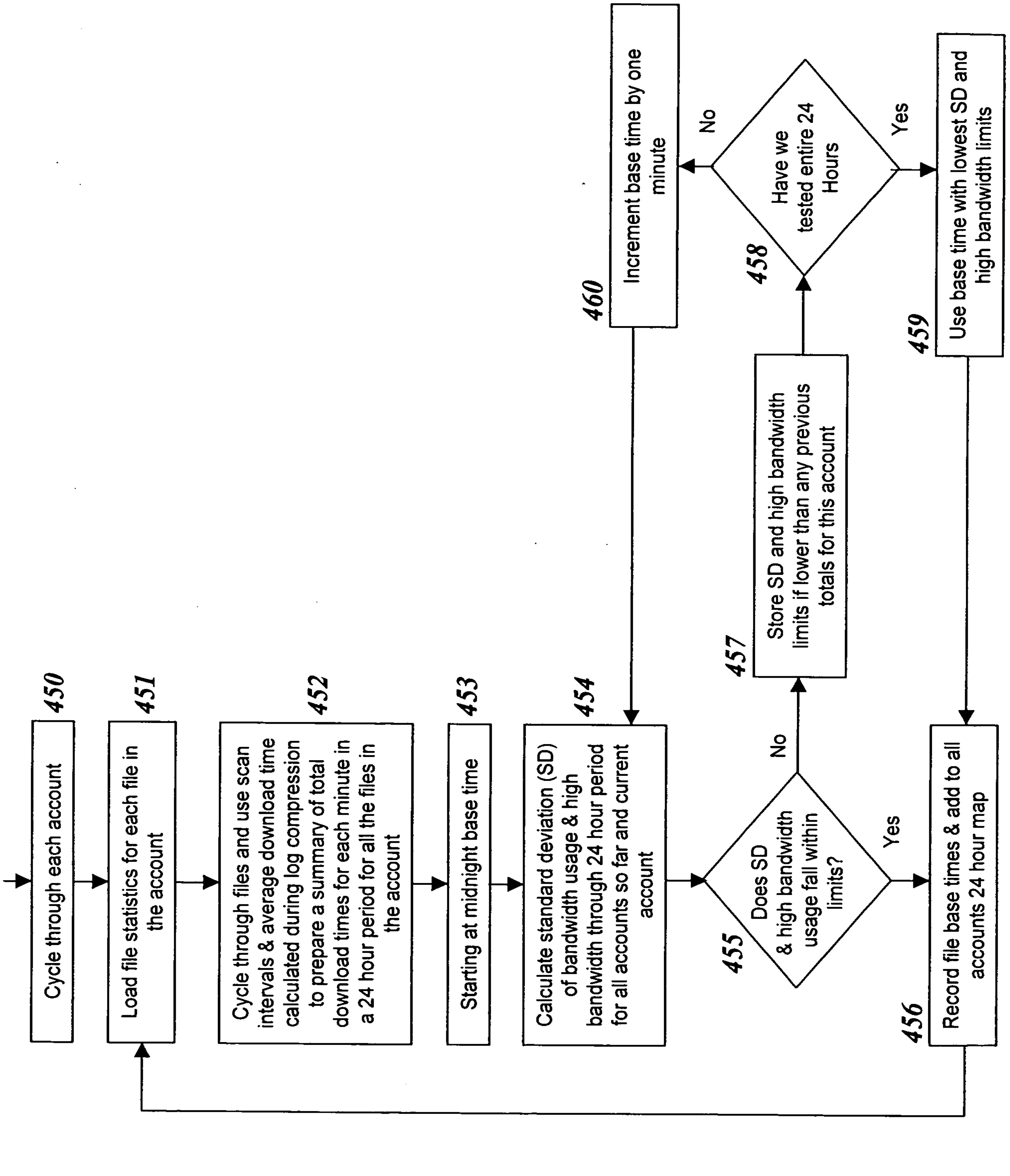
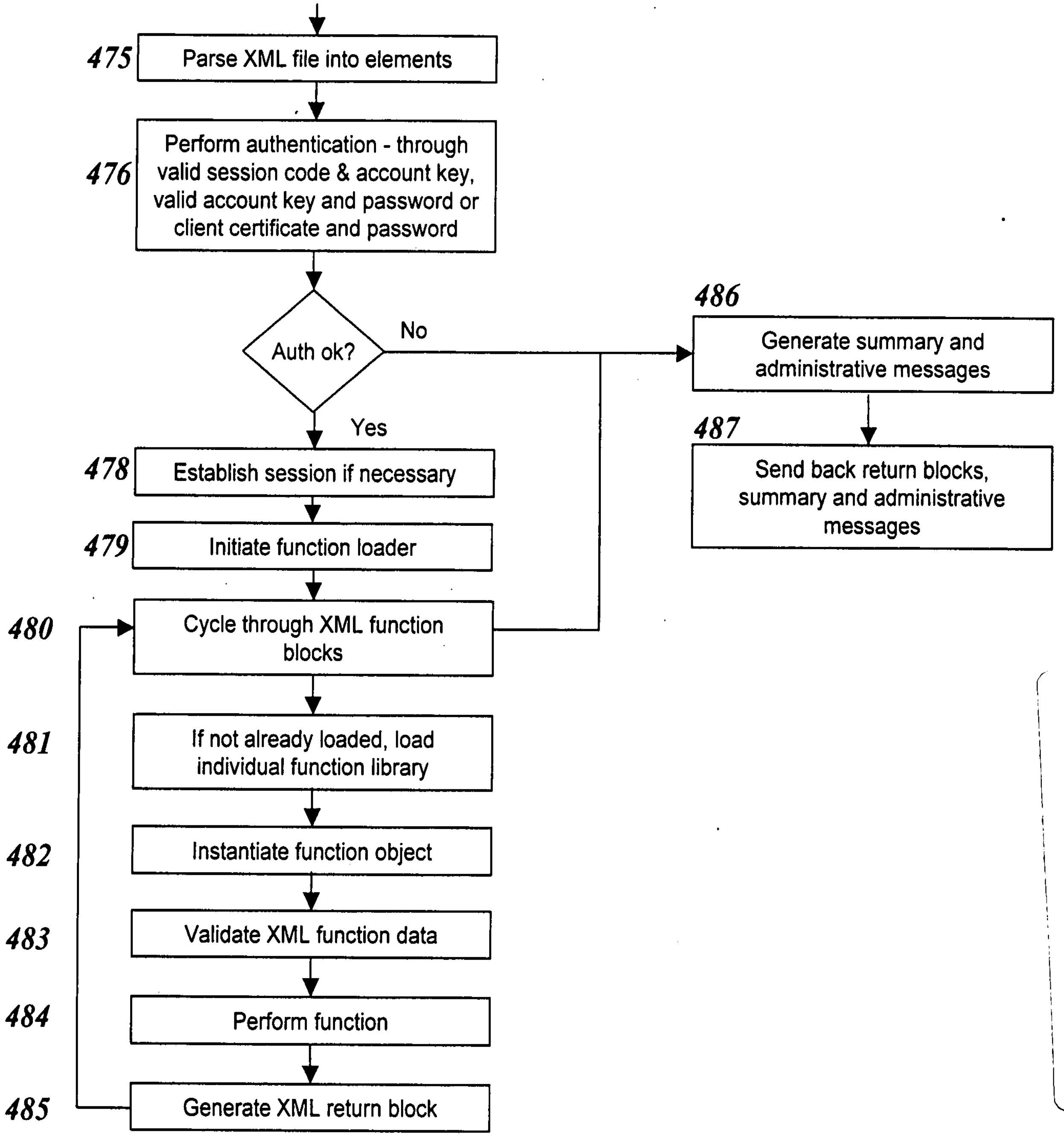
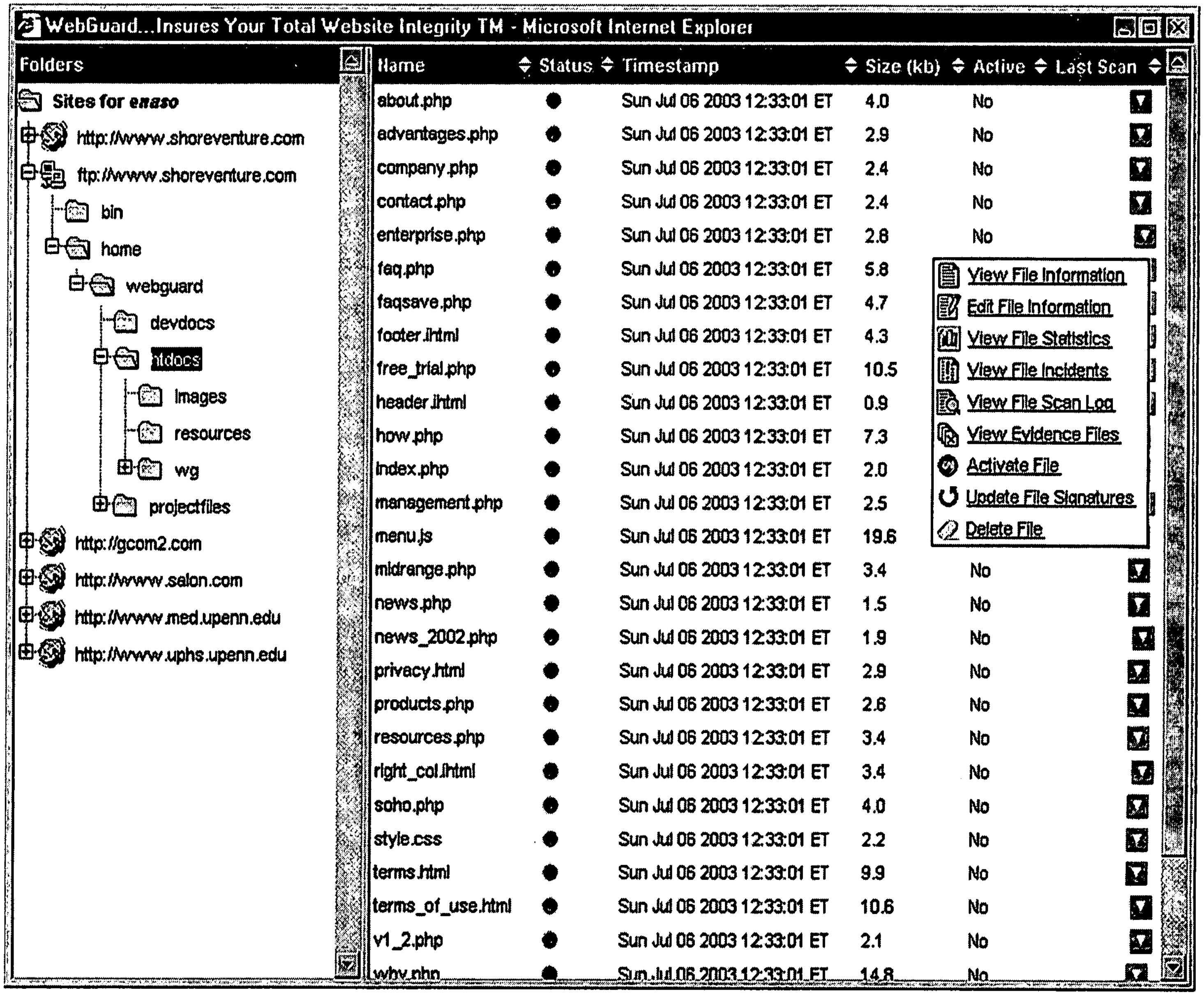

450
Cycle through each account
451
Load file statistics for each file in the account
452
Cycle through files and use scan intervals & average download time calculated during log compression to prepare a summary of total download times for each minute in a 24 hour period for all the files in the account
453
Starting at midnight base time
454
Calculate standard deviation (SD) of bandwidth usage & high bandwidth through 24 hour period for all accounts so far and current account
455
Does SD & high bandwidth usage fall within limits?
No
Yes
456
Record file base times & add to all accounts 24 hour map
457
Store SD and high bandwidth limits if lower than any previous totals for this account
458
Have we tested entire 24 Hours
No
Yes
459
Use base time with lowest SD and high bandwidth limits
460
Increment base time by one minute

Figure 21: XML Bridge (using XML-RPC Spec)


475
Parse XML file into elements
476
Perform authentication - through valid session code & account key, valid account key and password or client certificate and password
Auth ok?
No
Yes
486
Generate summary and administrative messages
487
Send back return blocks, summary and administrative messages
478
Establish session if necessary
479
Initiate function loader
480
Cycle through XML function blocks
481
If not already loaded, load individual function library
482
Instantiate function object
483
Validate XML function data
484
Perform function
485
Generate XML return block

Applicants: Matthew J. Dye, et al.
Filed: November 20, 2003 Customer No. 03000
For: REMOTE WEB SITE SECURITY SYSTEM
Docket No.: W1115/20001 Sheet 22 of 30

Figure 22: FileExplorer



Name	Status	Timestamp	Size (kb)	Active	Last Scan
about.php	●	Sun Jul 06 2003 12:33:01 ET	4.0	No	▼
advantages.php	●	Sun Jul 06 2003 12:33:01 ET	2.9	No	▼
company.php	●	Sun Jul 06 2003 12:33:01 ET	2.4	No	▼
contact.php	●	Sun Jul 06 2003 12:33:01 ET	2.4	No	▼
enterprise.php	●	Sun Jul 06 2003 12:33:01 ET	2.8	No	▼
faq.php	●	Sun Jul 06 2003 12:33:01 ET	5.8		
faqsave.php	●	Sun Jul 06 2003 12:33:01 ET	4.7		
footer.ihtml	●	Sun Jul 06 2003 12:33:01 ET	4.3		
free_trial.php	●	Sun Jul 06 2003 12:33:01 ET	10.5		
header.ihtml	●	Sun Jul 06 2003 12:33:01 ET	0.9		
how.php	●	Sun Jul 06 2003 12:33:01 ET	7.3		
index.php	●	Sun Jul 06 2003 12:33:01 ET	2.0		
management.php	●	Sun Jul 06 2003 12:33:01 ET	2.5		
menu.js	●	Sun Jul 06 2003 12:33:01 ET	19.6		
midrange.php	●	Sun Jul 06 2003 12:33:01 ET	3.4	No	▼
news.php	●	Sun Jul 06 2003 12:33:01 ET	1.5	No	▼
news_2002.php	●	Sun Jul 06 2003 12:33:01 ET	1.9	No	▼
privacy.html	●	Sun Jul 06 2003 12:33:01 ET	2.9	No	▼
products.php	●	Sun Jul 06 2003 12:33:01 ET	2.6	No	▼
resources.php	●	Sun Jul 06 2003 12:33:01 ET	3.4	No	▼
right_col.ihtml	●	Sun Jul 06 2003 12:33:01 ET	3.4	No	▼
soho.php	●	Sun Jul 06 2003 12:33:01 ET	4.0	No	▼
style.css	●	Sun Jul 06 2003 12:33:01 ET	2.2	No	▼
terms.html	●	Sun Jul 06 2003 12:33:01 ET	9.9	No	▼
terms_of_use.html	●	Sun Jul 06 2003 12:33:01 ET	10.6	No	▼
v1_2.php	●	Sun Jul 06 2003 12:33:01 ET	2.1	No	▼
why.php	●	Sun Jul 06 2003 12:33:01 ET	14.8	No	▼

Figure 23: WebGuard OnSite Architecture

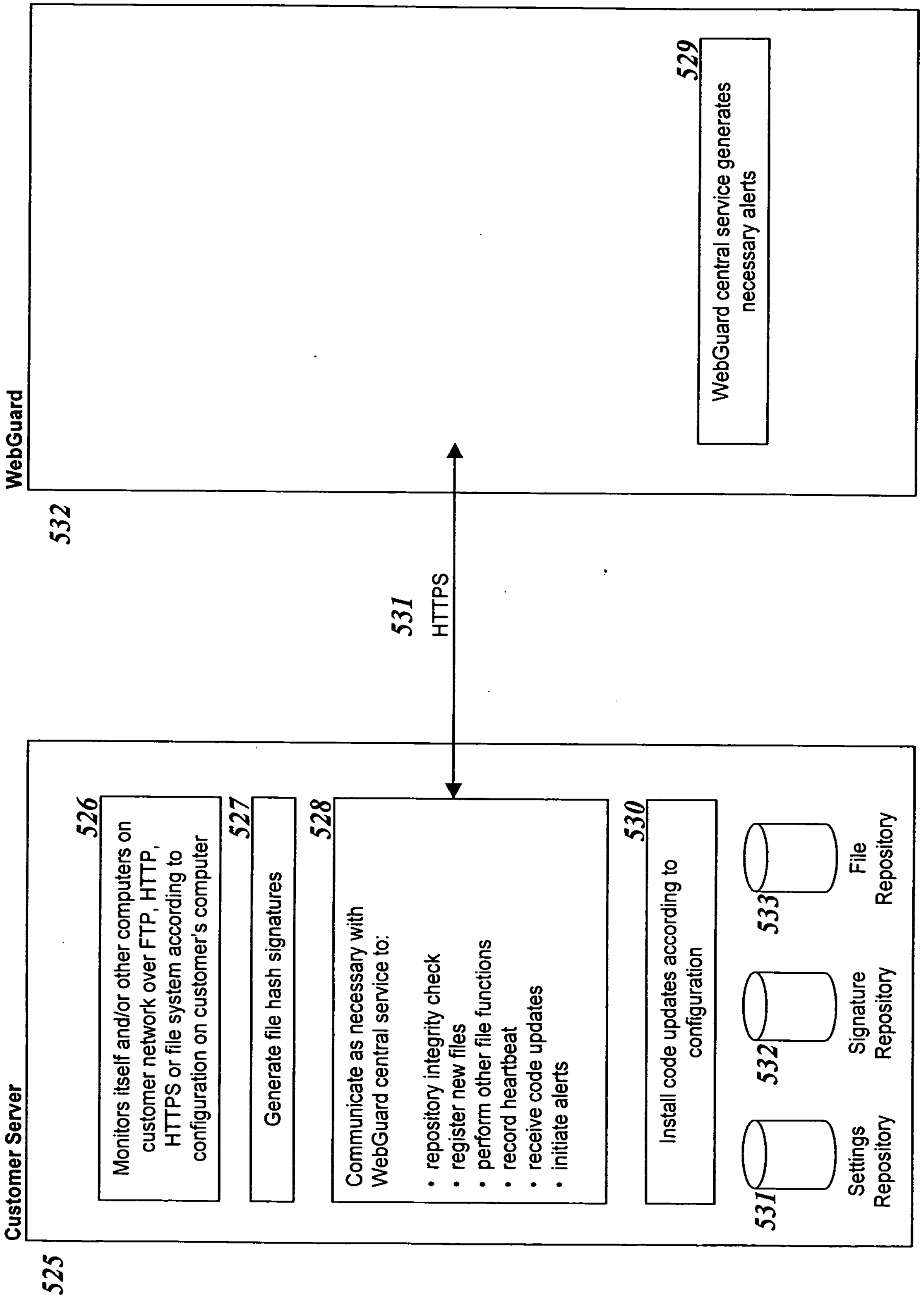
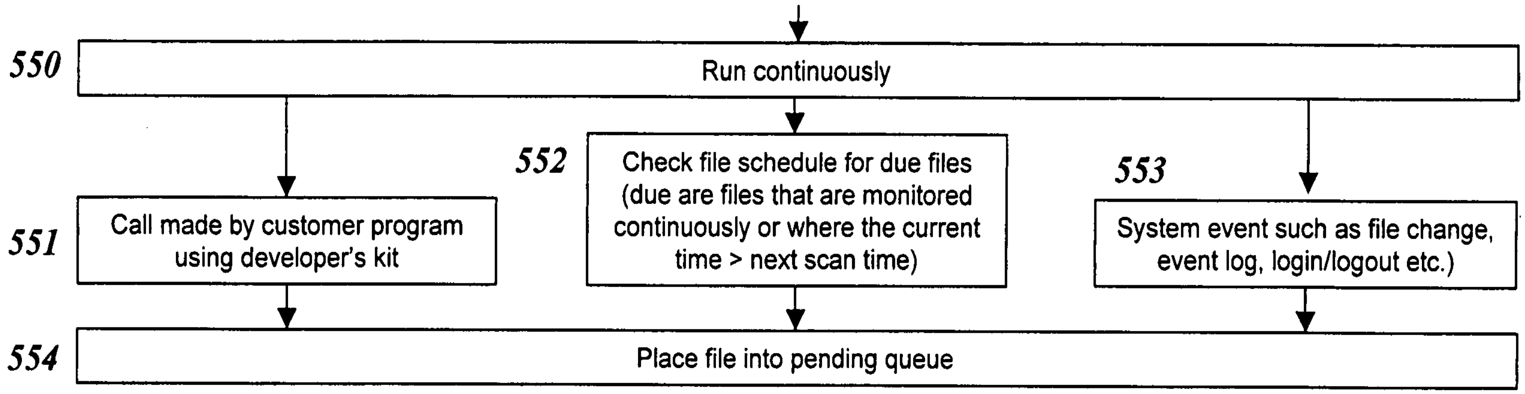

525
Customer Server
526
Monitors itself and/or other computers on customer network over FTP, HTTP, HTTPS or file system according to configuration on customer's computer
527
Generate file hash signatures
528
Communicate as necessary with WebGuard central service to:
• repository integrity check
• register new files
• perform other file functions
• record heartbeat
• receive code updates
• initiate alerts
530
Install code updates according to configuration
531
Settings Repository
532
Signature Repository
533
File Repository
531
HTTPS
532
WebGuard
529
WebGuard central service generates necessary alerts

Figure 24: OnSite File Scheduling (on customer server)


550
Run continuously
551
Call made by customer program using developer's kit
552
Check file schedule for due files (due are files that are monitored continuously or where the current time > next scan time)
553
System event such as file change, event log, login/logout etc.)
554
Place file into pending queue

Applicants: Matthew J. Dye, et al.
Filed: November 20, 2003 Customer No. 03000
For: REMOTE WEB SITE SECURITY SYSTEM
Docket No.: W1115/20001 Sheet 25 of 30

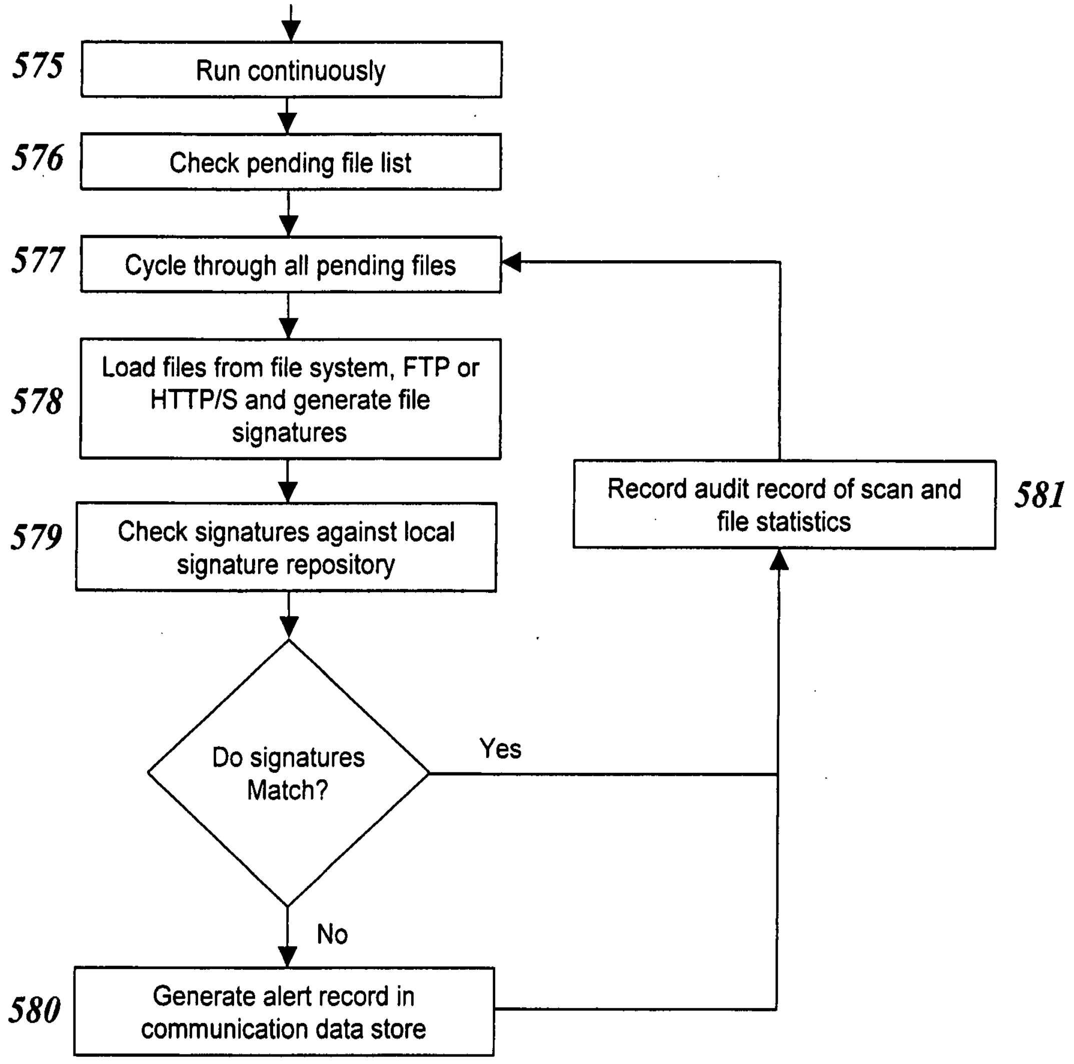

575
Run continuously
576
Check pending file list
577
Cycle through all pending files
578
Load files from file system, FTP or HTTP/S and generate file signatures
579
Check signatures against local signature repository
Do signatures Match?
Yes
No
580
Generate alert record in communication data store
Record audit record of scan and file statistics
581

Figure 25: OnSite File Scanning (on customer server)

Figure 26: OnSite File Communications (on customer server)

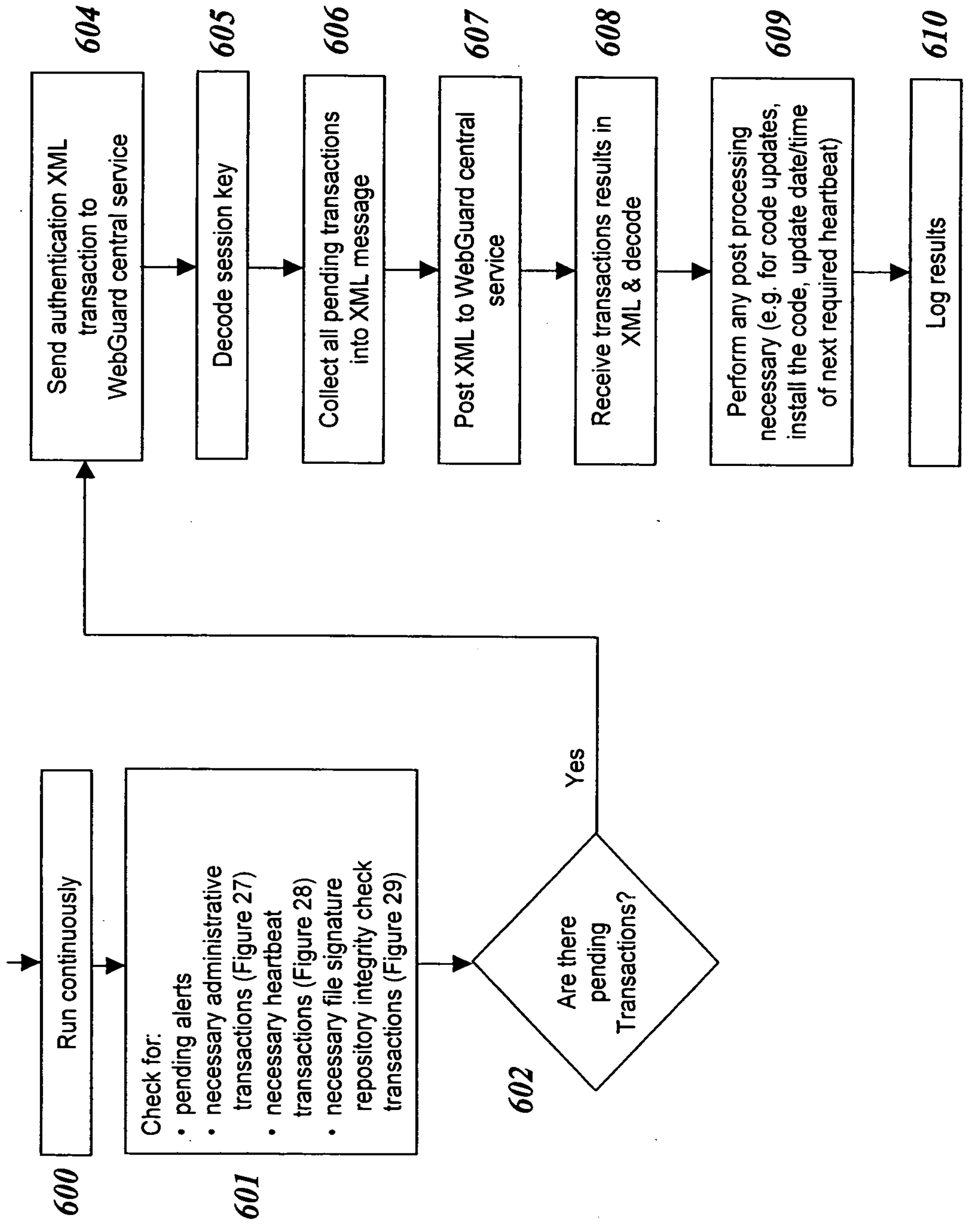

600
Run continuously
601
Check for:
• pending alerts
• necessary administrative transactions (Figure 27)
• necessary heartbeat transactions (Figure 28)
• necessary file signature repository integrity check transactions (Figure 29)
602
Are there pending Transactions?
Yes
604
Send authentication XML transaction to WebGuard central service
605
Decode session key
606
Collect all pending transactions into XML message
607
Post XML to WebGuard central service
608
Receive transactions results in XML & decode
609
Perform any post processing necessary (e.g. for code updates, install the code, update date/time of next required heartbeat)
610
Log results

Figure 27: OnSite File Administrative Transactions

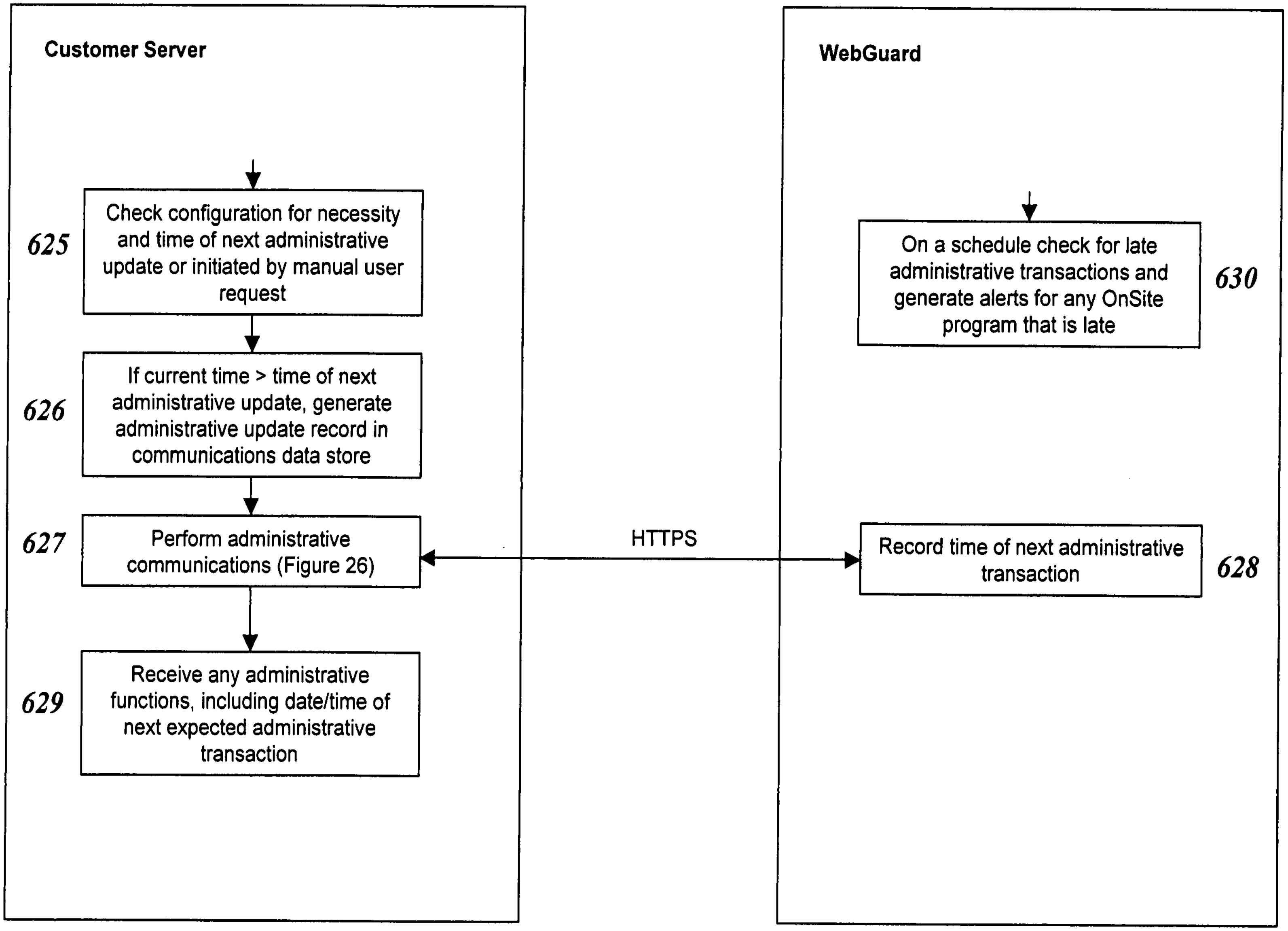
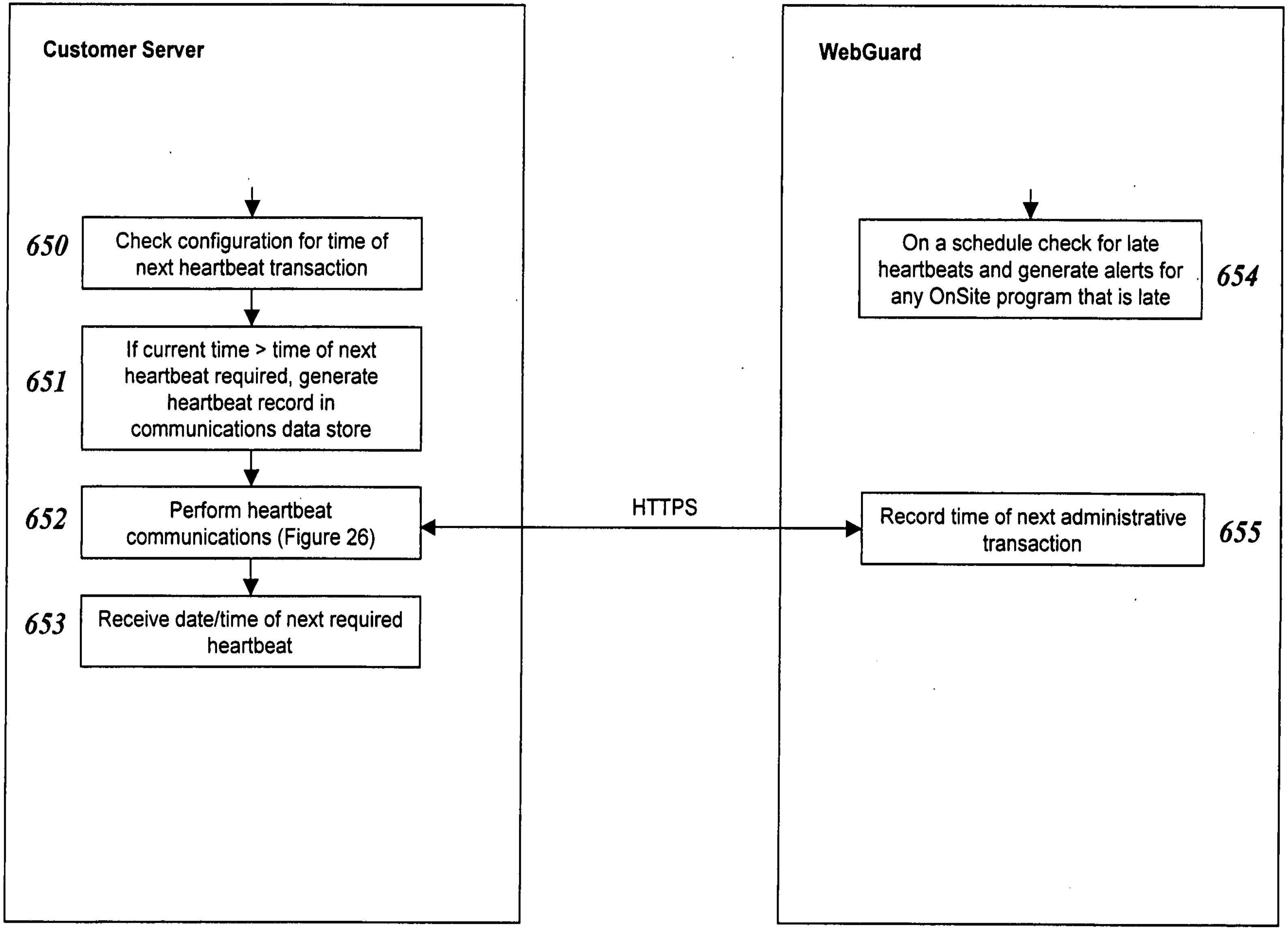
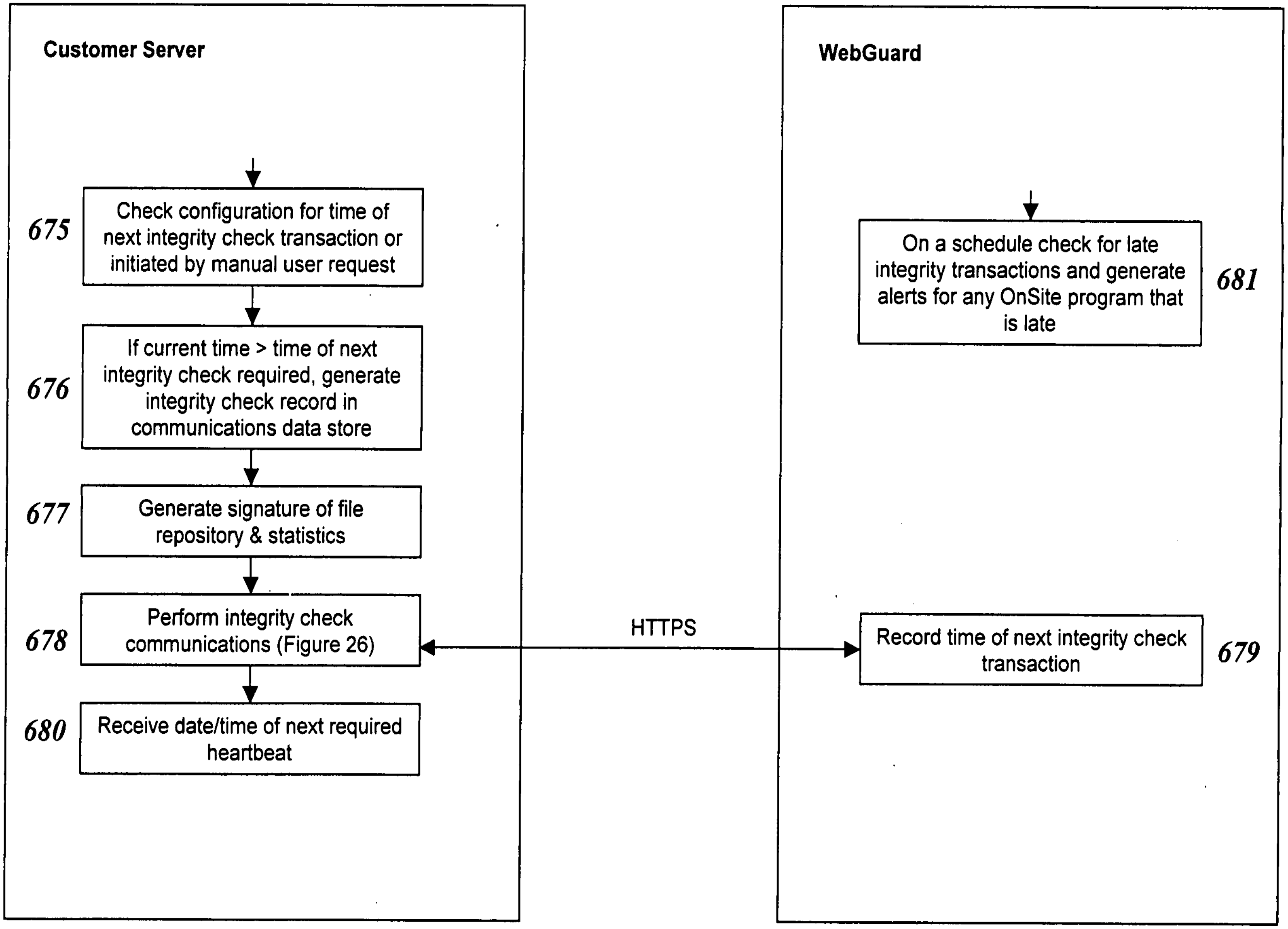

Customer Server
625
Check configuration for necessity and time of next administrative update or initiated by manual user request
626
If current time > time of next administrative update, generate administrative update record in communications data store
627
Perform administrative communications (Figure 26)
629
Receive any administrative functions, including date/time of next expected administrative transaction
HTTPS
WebGuard
On a schedule check for late administrative transactions and generate alerts for any OnSite program that is late
630
Record time of next administrative transaction
628

Figure 28: OnSite Heartbeat Transactions


Customer Server
650
Check configuration for time of next heartbeat transaction
651
If current time > time of next heartbeat required, generate heartbeat record in communications data store
652
Perform heartbeat communications (Figure 26)
653
Receive date/time of next required heartbeat
HTTPS
WebGuard
On a schedule check for late heartbeats and generate alerts for any OnSite program that is late
654
Record time of next administrative transaction
655

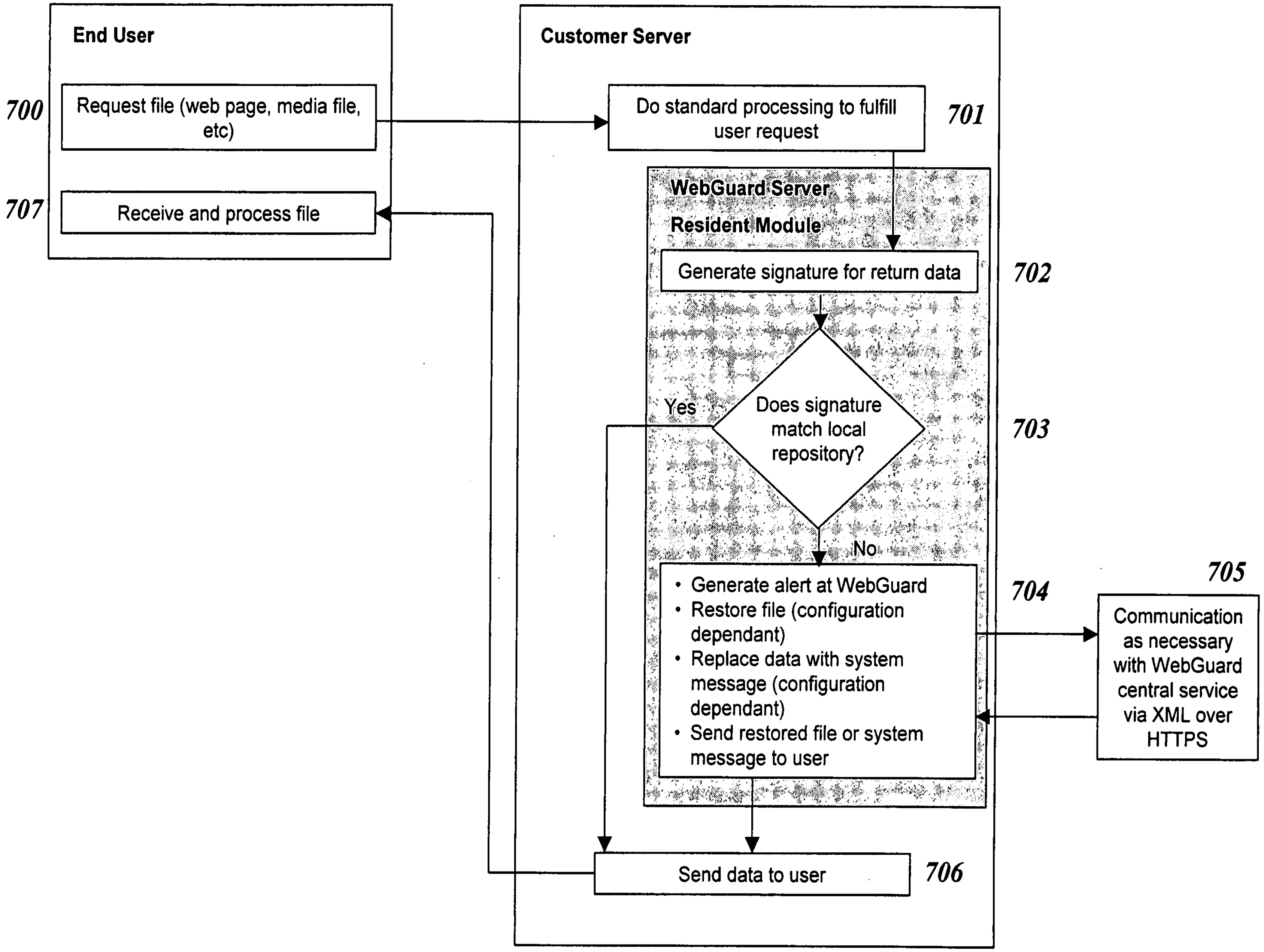
Note: File repository integrity checks and alerts generated with WebGuard central service also count as heartbeats

Figure 29: OnSite Local File Repository Integrity Check Transactions


Customer Server
675
Check configuration for time of next integrity check transaction or initiated by manual user request
676
If current time > time of next integrity check required, generate integrity check record in communications data store
677
Generate signature of file repository & statistics
678
Perform integrity check communications (Figure 26)
680
Receive date/time of next required heartbeat
HTTPS
WebGuard
On a schedule check for late integrity transactions and generate alerts for any OnSite program that is late
681
Record time of next integrity check transaction
679

Note: File repository integrity checks and alerts generated with WebGuard central service also count as heartbeats

Figure 30: OnSite ServerGuard


End User
700
Request file (web page, media file, etc)
707
Receive and process file
Customer Server
Do standard processing to fulfill user request
701
WebGuard Server Resident Module
Generate signature for return data
702
Does signature match local repository?
703
Yes
No
• Generate alert at WebGuard
• Restore file (configuration dependant)
• Replace data with system message (configuration dependant)
• Send restored file or system message to user
704
705
Communication as necessary with WebGuard central service via XML over HTTPS
Send data to user
706